OIPE
JUL 15 2004
PATENT & TRADEMARK OFFICE

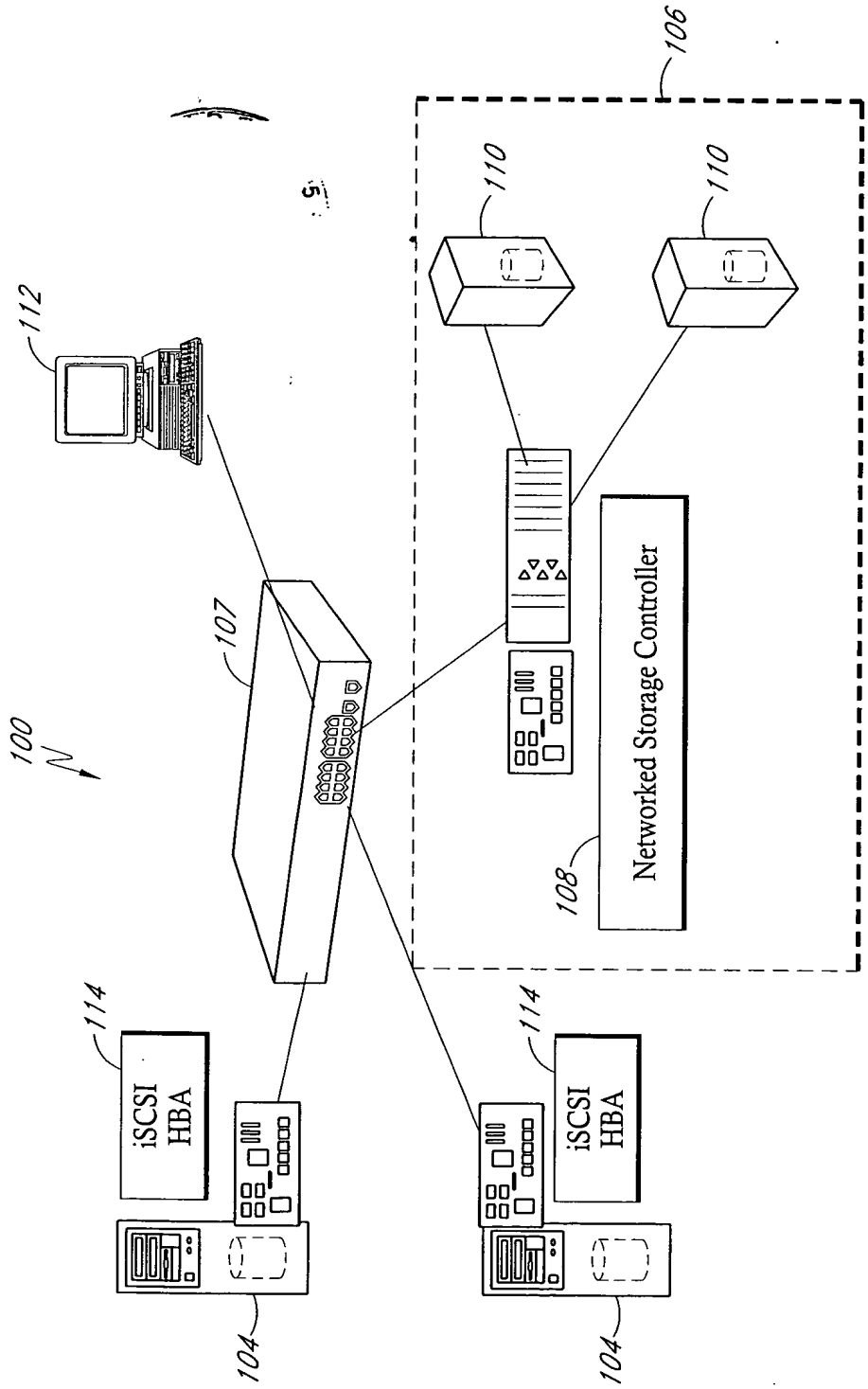


FIG. 1

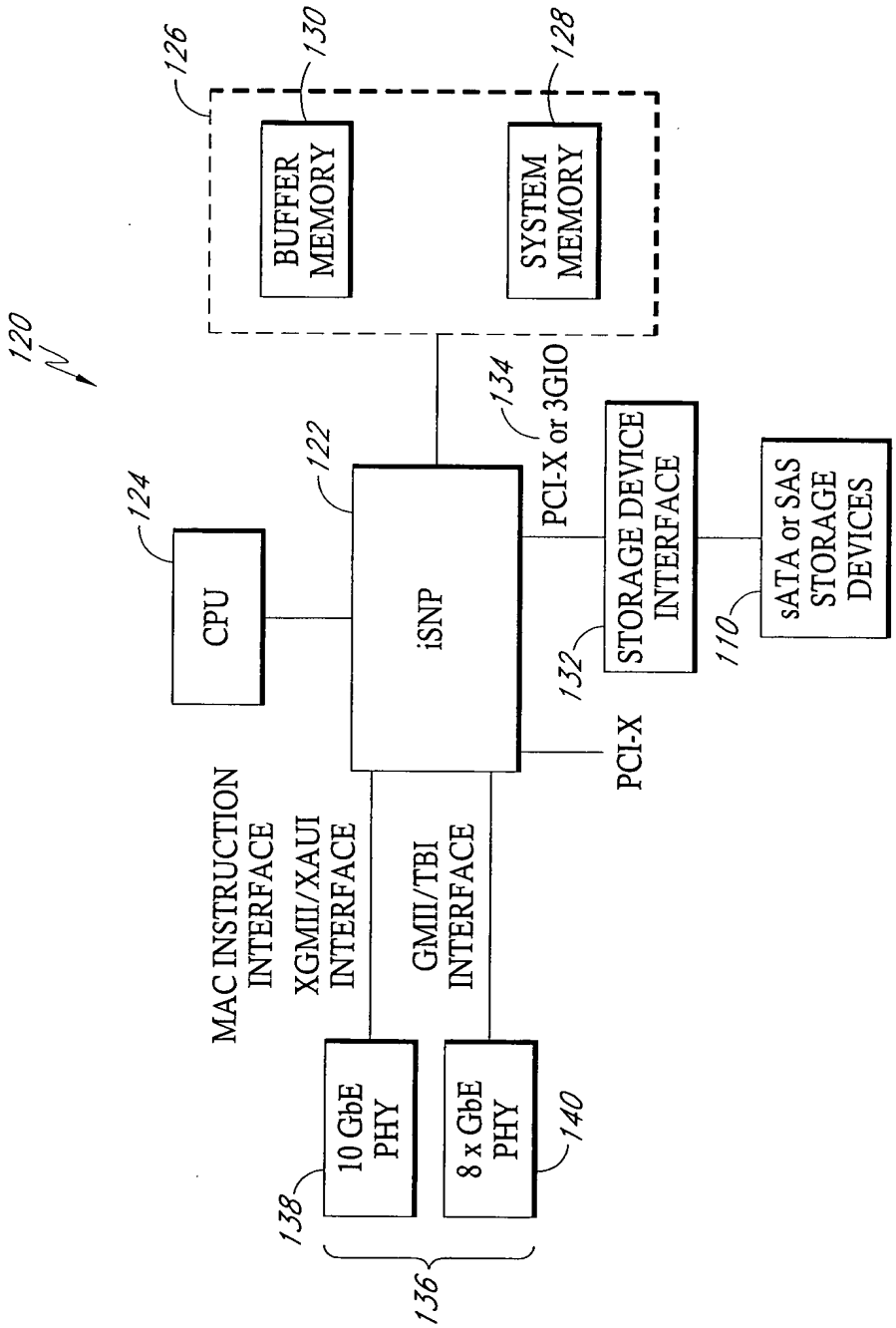
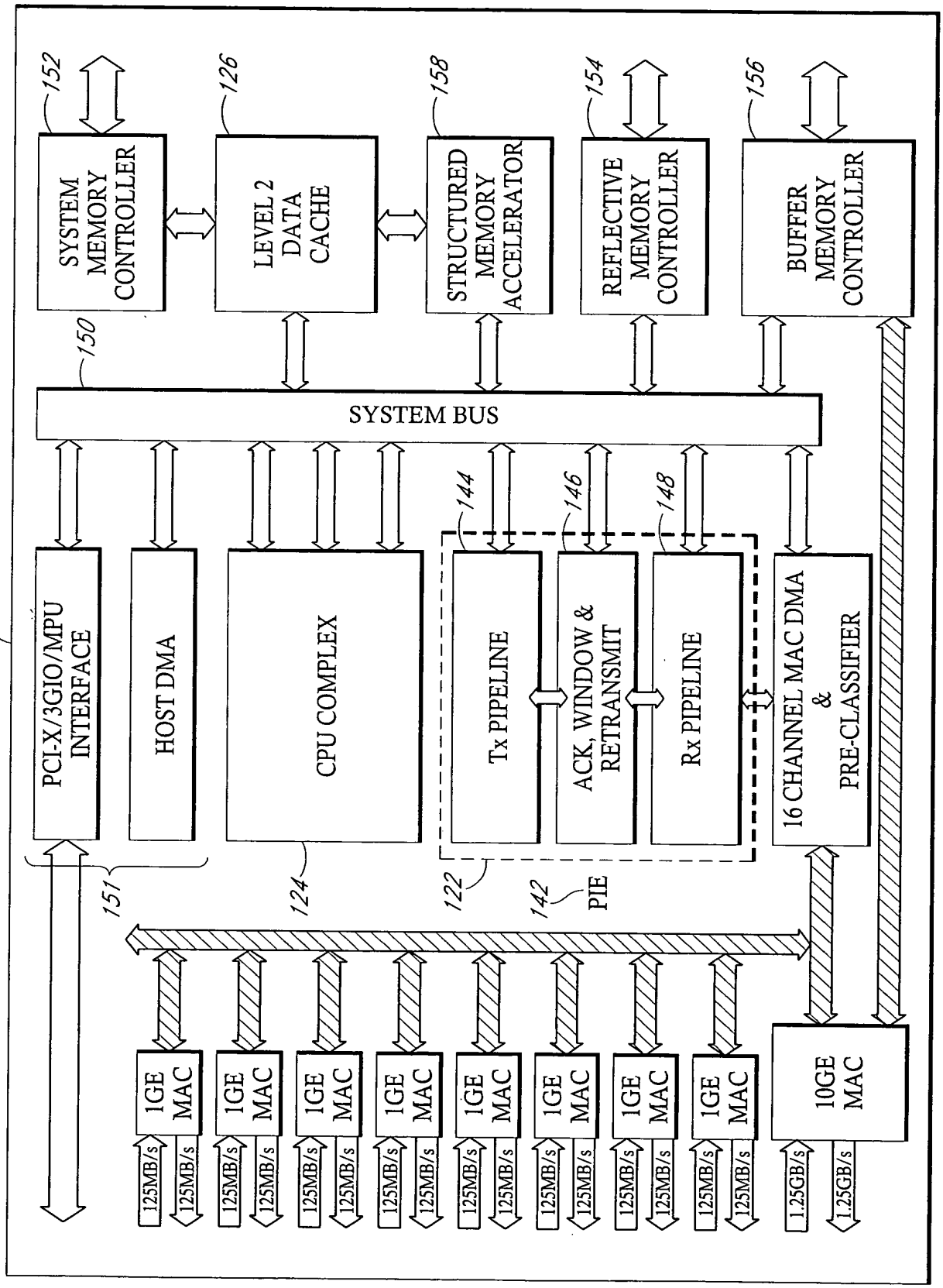

120
126
130
128
BUFFER MEMORY
SYSTEM MEMORY
134
PCI-X or 3GIO
122
124
CPU
iSNP
STORAGE DEVICE INTERFACE
110
sATA or SAS STORAGE DEVICES
132
PCI-X
MAC INSTRUCTION INTERFACE
XGMII/XAUI INTERFACE
GMII/TBI INTERFACE
10 GbE PHY
8 x GbE PHY
140
138
136

FIG. 2

FIG. 3A


120
152
126
158
154
156
SYSTEM MEMORY CONTROLLER
LEVEL 2 DATA CACHE
STRUCTURED MEMORY ACCELERATOR
REFLECTIVE MEMORY CONTROLLER
BUFFER MEMORY CONTROLLER
150
SYSTEM BUS
144
146
148
PCI-X/3GIO/MPU INTERFACE
HOST DMA
CPU COMPLEX
Tx PIPELINE
ACK, WINDOW & RETRANSMIT
Rx PIPELINE
16 CHANNEL MAC DMA & PRE-CLASSIFIER
151
124
122
142
PIE
1GE MAC
10GE MAC
125MB/s
1.25GB/s

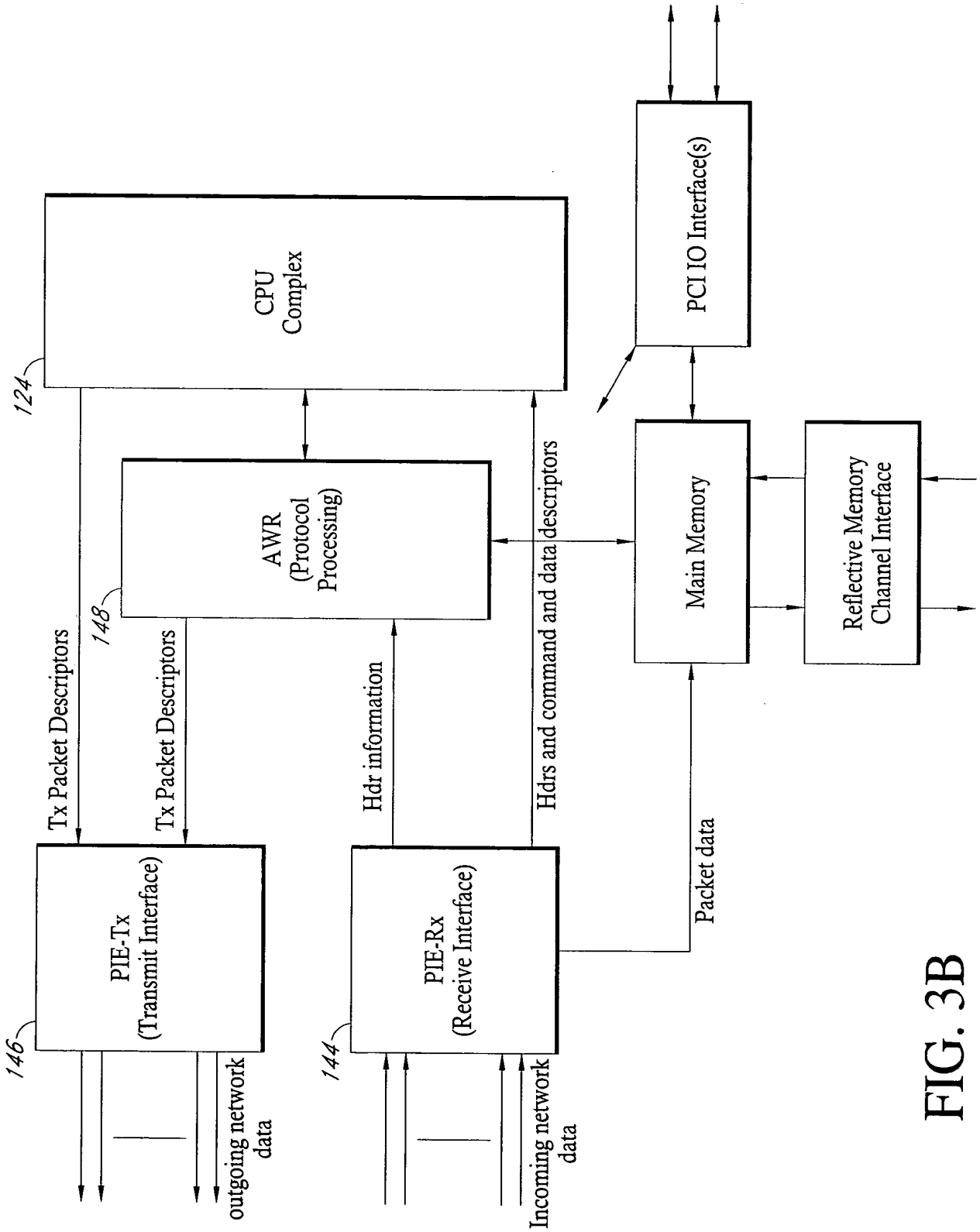

CPU Complex
124
PCI IO Interface(s)
AWR (Protocol Processing)
148
Main Memory
Reflective Memory Channel Interface
Tx Packet Descriptors
Tx Packet Descriptors
Hdr information
Hdrs and command and data descriptors
Packet data
PIE-Tx (Transmit Interface)
146
PIE-Rx (Receive Interface)
144
outgoing network data
Incoming network data

FIG. 3B

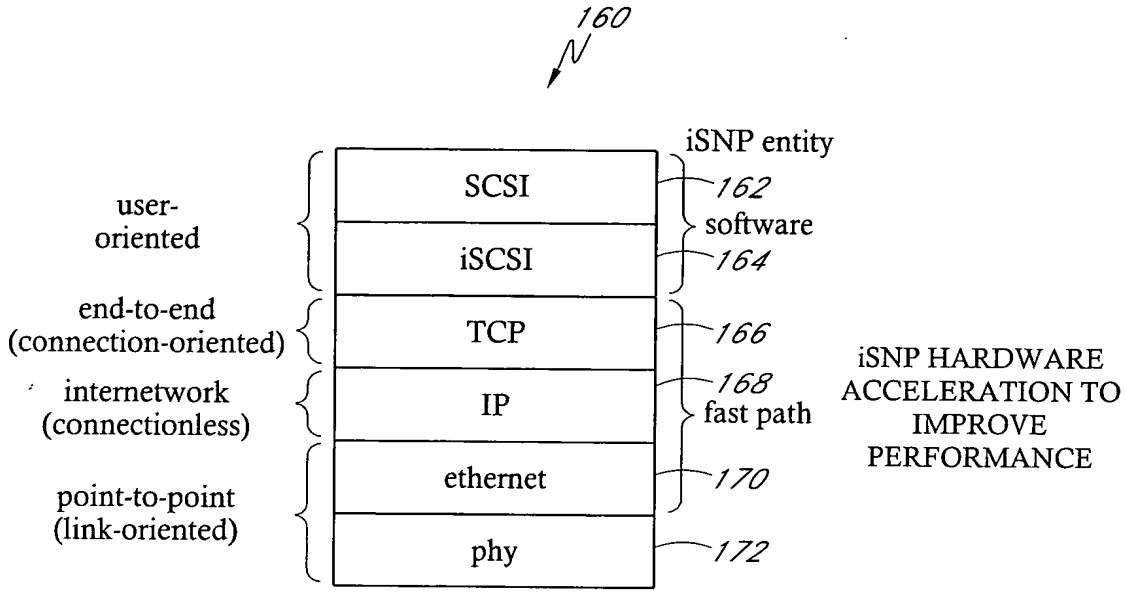

160
iSNP entity
SCSI
iSCSI
TCP
IP
ethernet
phy
162
software
164
166
168
fast path
170
172
user-oriented
end-to-end (connection-oriented)
internetwork (connectionless)
point-to-point (link-oriented)
iSNP HARDWARE ACCELERATION TO IMPROVE PERFORMANCE

FIG. 4A

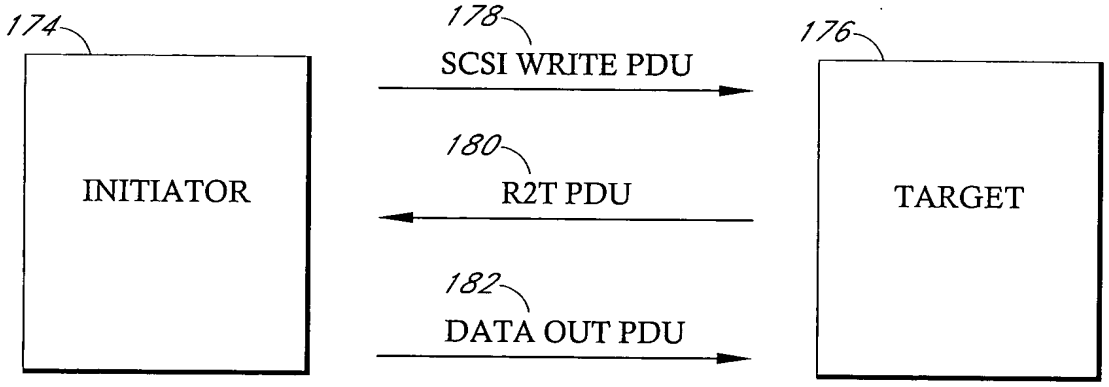

174
INITIATOR
178
SCSI WRITE PDU
180
R2T PDU
182
DATA OUT PDU
176
TARGET

FIG. 4B

segment	iSCSI PDU			
BHS (basic header segment, bytes 0-47)	B0[6]=I (immed delivery) B0[5:0]=opcode, initiator target 'h01=SCSI command 'h21=SCSI response 'h02=SCSI task mgmt req 'h22=SCSI task mgmt rsp 'h03=login command 'h23-=login response 'h05=SCSI data out 'h25=SCSI data in 'h06=logout command 'h26=logout response 'h10=SNACK (seq# ack) req 'h31=R2T (rdy to xfr) 'h31'=reject		B1-B3-opcode specific fields SCSI command: B1[7]=F (final), B1[6]=R (read), B1[5]=W (write) B1[2:0]=task attrib (0=untagged, 1=simple, 2=ord, 3=hoq, 4=aca) B3=CRN (command reference number) SCSI data out: B1[7]=F SCSI data in: B1[7]=F, B1[2]=o (overflow), B1[1]=U (underflow), B1[0]=S (status, if set: B3=status, B20-23=residual count, B24-27=statusSN) SCSI response: B1[2]=O, B1[1]=U, B1[4]=O for bidir read, B1[3]=U bi rd B2=response (0=cmd complete, 1=targ fail)	
	B4=total AHS length		B5-7=data segment length (not including pad)	
	B8-15=logical unit number (LUN) (64 bits)			
	B16-19=initiator task tag			
	B20-47=opcode specific fields (7 words) SCSI command:B20-23=expected data length, B24-27=cmdSN (cmd seq number), B28-31=expStatSN (acks status thru SN-1) B32-47=CDB (16 bytes) R2T: B20-23=targ transfer tag, B24-27=statSN (for next status), B28-31=expCmdSN (cmd ack to init), B32-35=maxCmdSN B36-39=R2TSN (R2T PDU number, starts at 0), B40-43=buffer offset, B44-47=desired data length (in bytes) SCSI data out: B20-23=targ transfer tag from R2T (or F's). B28-31=expStatSN (expected status sequence number) B36-39=dataSN, B40-43=buffer offset (for this payload relative to complete data transfer) SCSI data in: B20-23=residual count, B24-27=statSN, B28-35=maxCmdSN, B36-39=dtaSN, B40-43=buffer offset SCSI response: B24-27=statSN, B28-31=expCmdSN, B36-39=expDataSN, B40-43=bidirectional read residual count, B44-47=residual count			
ABS (additional header seg. optional)	B0=B1=AHS length		B2=AHS type B2[7]=drop B2[5:0]=ahs code 1=extended CDB 2=exp bidir read length	B3=AHS specific
	extended CDB: B4..n=extended CDB (if necessary pad to full word) bidir read length: B4=expected read data length			
hdr digest (optional)	CRC for header segment(s)			
data seg. (optional)	Data (if necessary pad to full word)			
data digest (optional)	CRC for data segment			

FIG. 5A

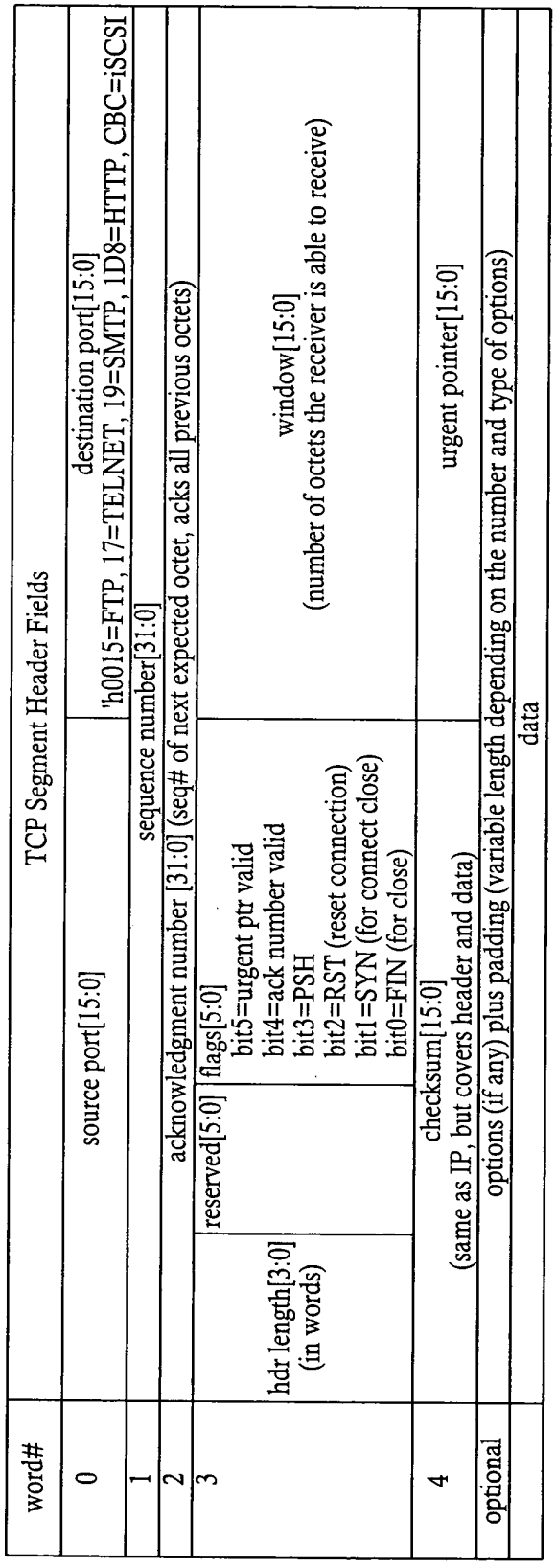

TCP Segment Header Fields
word#
0
source port[15:0]
destination port[15:0]
'h0015=FTP, 17=TELNET, 19=SMTP, 1D8=HTTP, CBC=iSCSI
1
sequence number[31:0]
2
acknowledgment number [31:0] (seq# of next expected octet, acks all previous octets)
3
hdr length[3:0] (in words)
reserved[5:0]
flags[5:0]
bit5=urgent ptr valid
bit4=ack number valid
bit3=PSH
bit2=RST (reset connection)
bit1=SYN (for connect close)
bit0=FIN (for close)
window[15:0]
(number of octets the receiver is able to receive)
4
checksum[15:0]
(same as IP, but covers header and data)
urgent pointer[15:0]
optional
options (if any) plus padding (variable length depending on the number and type of options)
data

FIG. 5B

word#	IP Packet Header Fields				
0	version[3:0]	IHL[3:0] (header len in words, min 5=no options)	TOS[7:0] (type of service, specifics precedence, delay, throughput and reliability parameters)	TLEN[15:0] (total length including header, in octets)	
1	ID[15:0] (with src addr, dst addr, and user protocol uniquely identifies the IP datagram)			flag[2:0] flag{1]=dont frag flag[0]=more frags	fragment offset[12:0] (in 64-bit units)
2	TTL[7:0] (time to live, decremented at each hop, if 0 discard the datagram)		Protocol[7:0] 1-ICMP 6=TCP 17+UDP	Header checksum[15:0] (1's complement of the 1's complement sum of data interpreted 16 bits at at time, with end-around carry)	
3	Source IP address[31:0] (coded to allow a variable number of hits to specify the network and station)				
4	Destination IP address[31:0] (as for source address, 224-239.x.x.x=multicast)				
:	Options (if any) plus padding (variable length depending on the number and type of options)				
:	Data (multiple of 8 bits in length)				

FIG. 5C

ethernet frame field	pre-amble	SFD (start frame)	DA (dst addr)	SA (src addr)	type (ethernet II) or length of info (IEE 802.3)	information	FCS (frame chk seq)	extension
#octets	7	1	6	6	2	46-1500	4	
notes					'h0800=IP 'h0806=ARP 'h86dd=IPv6 'h8100=VLAN (insert w/2-byte tag before type/length) <='h05dc=length (rfc 1042, insert w/constant 0xaaaa03_000000 before type)	jumbo: max 9000 GE: min=512 for half duplex CSMA/CD) <min: add pad bytes		special nondata symbols for half duplex CSMA/CD

FIG. 5D

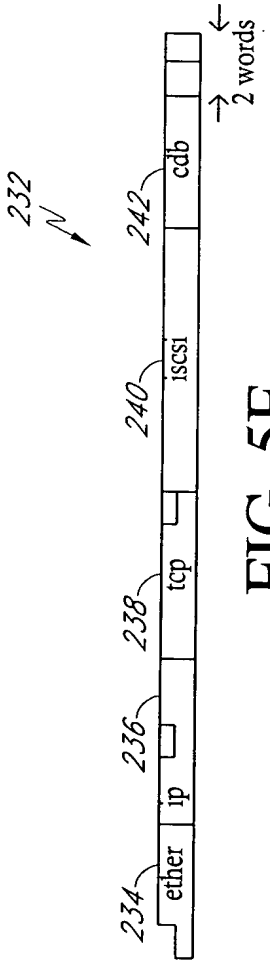


FIG. 5E

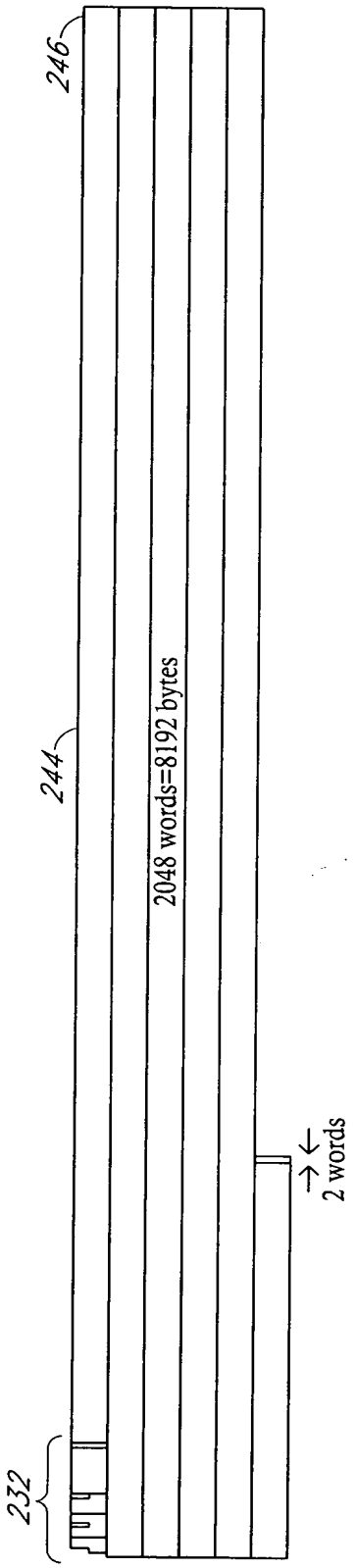


FIG. 5F

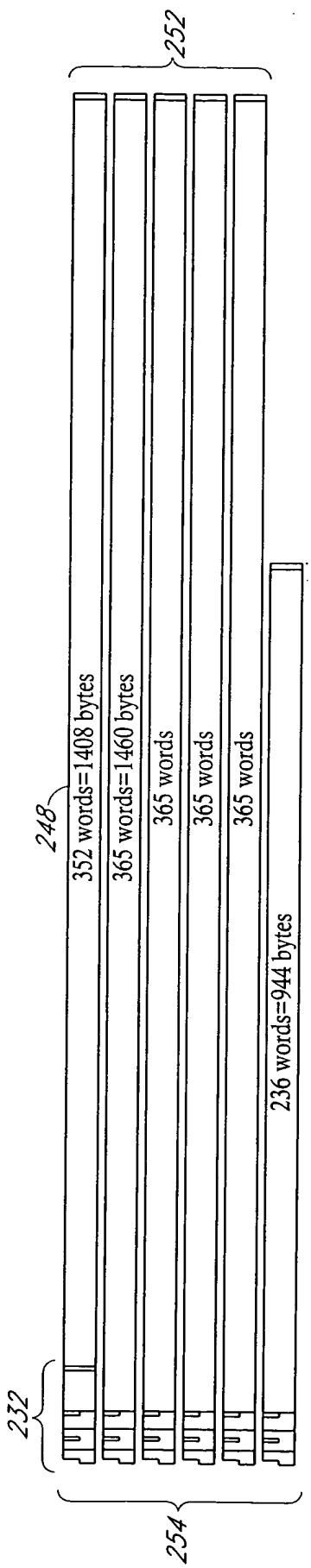


FIG. 5G

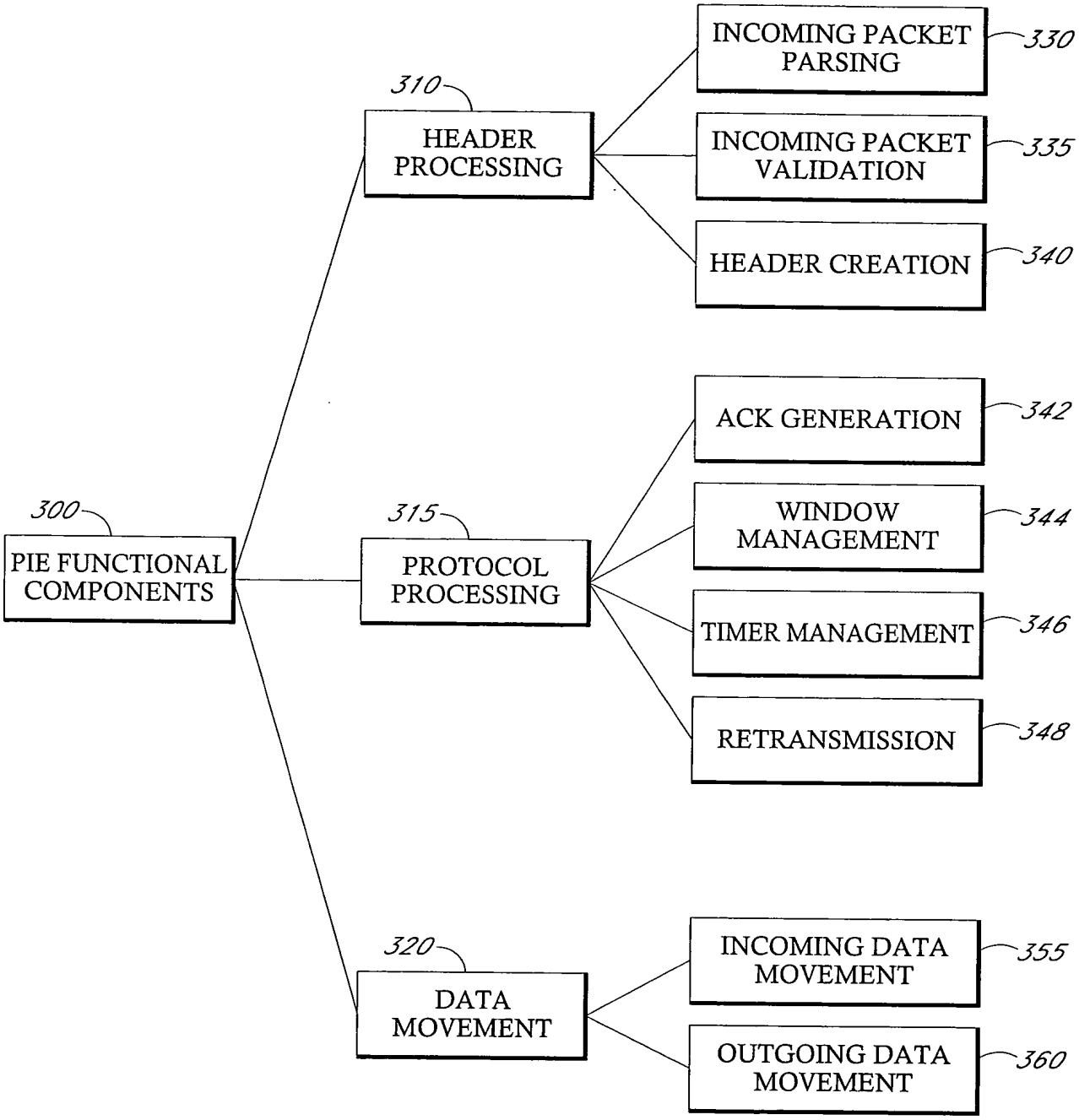

300
PIE FUNCTIONAL COMPONENTS
310
HEADER PROCESSING
315
PROTOCOL PROCESSING
320
DATA MOVEMENT
INCOMING PACKET PARSING
330
INCOMING PACKET VALIDATION
335
HEADER CREATION
340
ACK GENERATION
342
WINDOW MANAGEMENT
344
TIMER MANAGEMENT
346
RETRANSMISSION
348
INCOMING DATA MOVEMENT
355
OUTGOING DATA MOVEMENT
360

FIG. 6A

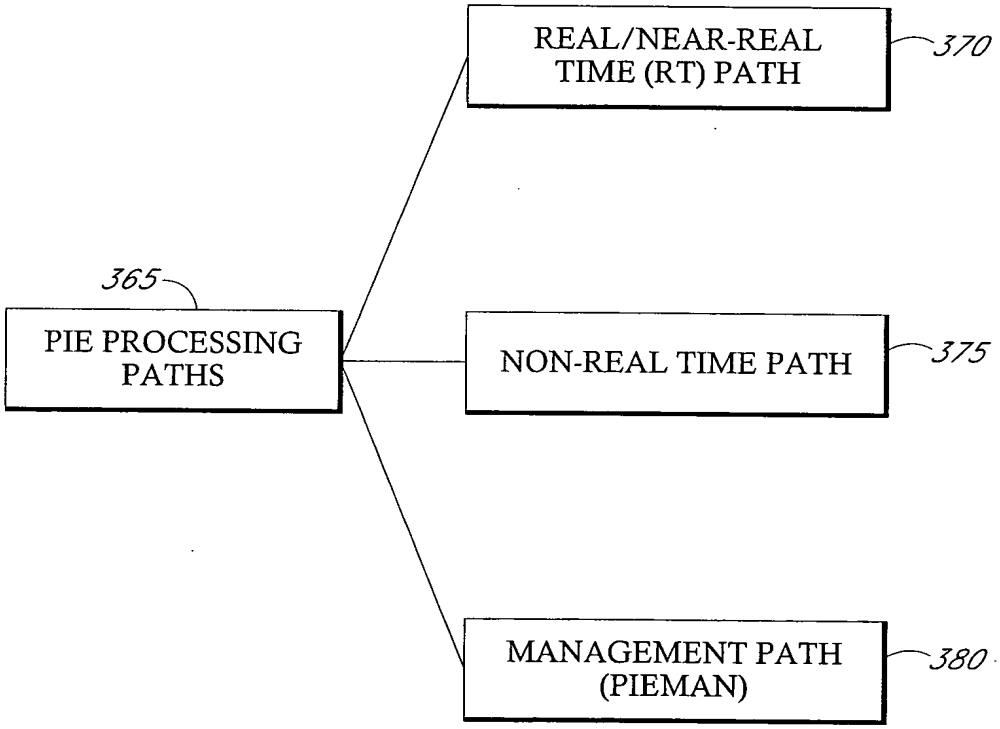

REAL/NEAR-REAL TIME (RT) PATH
370
365
PIE PROCESSING PATHS
NON-REAL TIME PATH
375
MANAGEMENT PATH (PIEMAN)
380

FIG. 6B

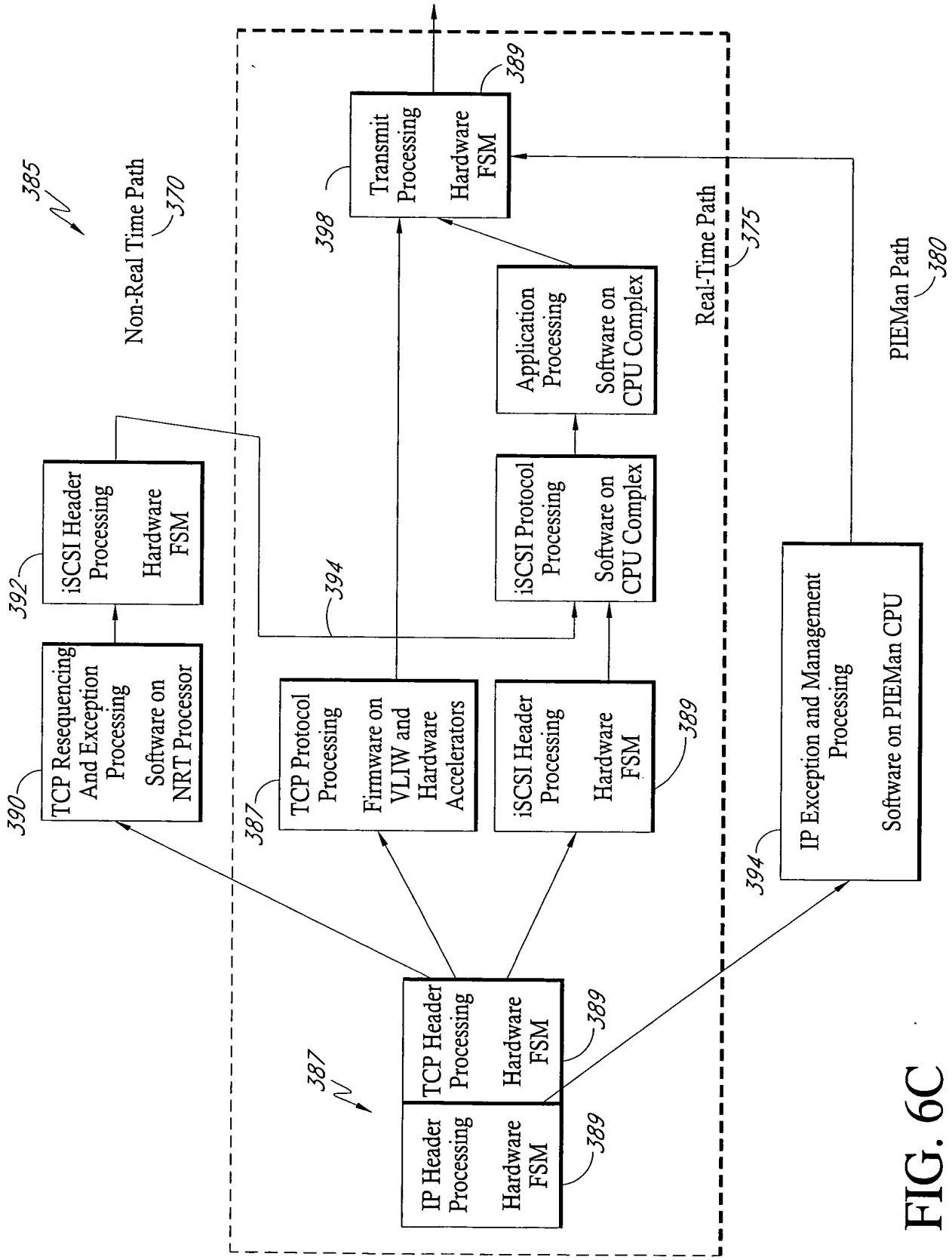
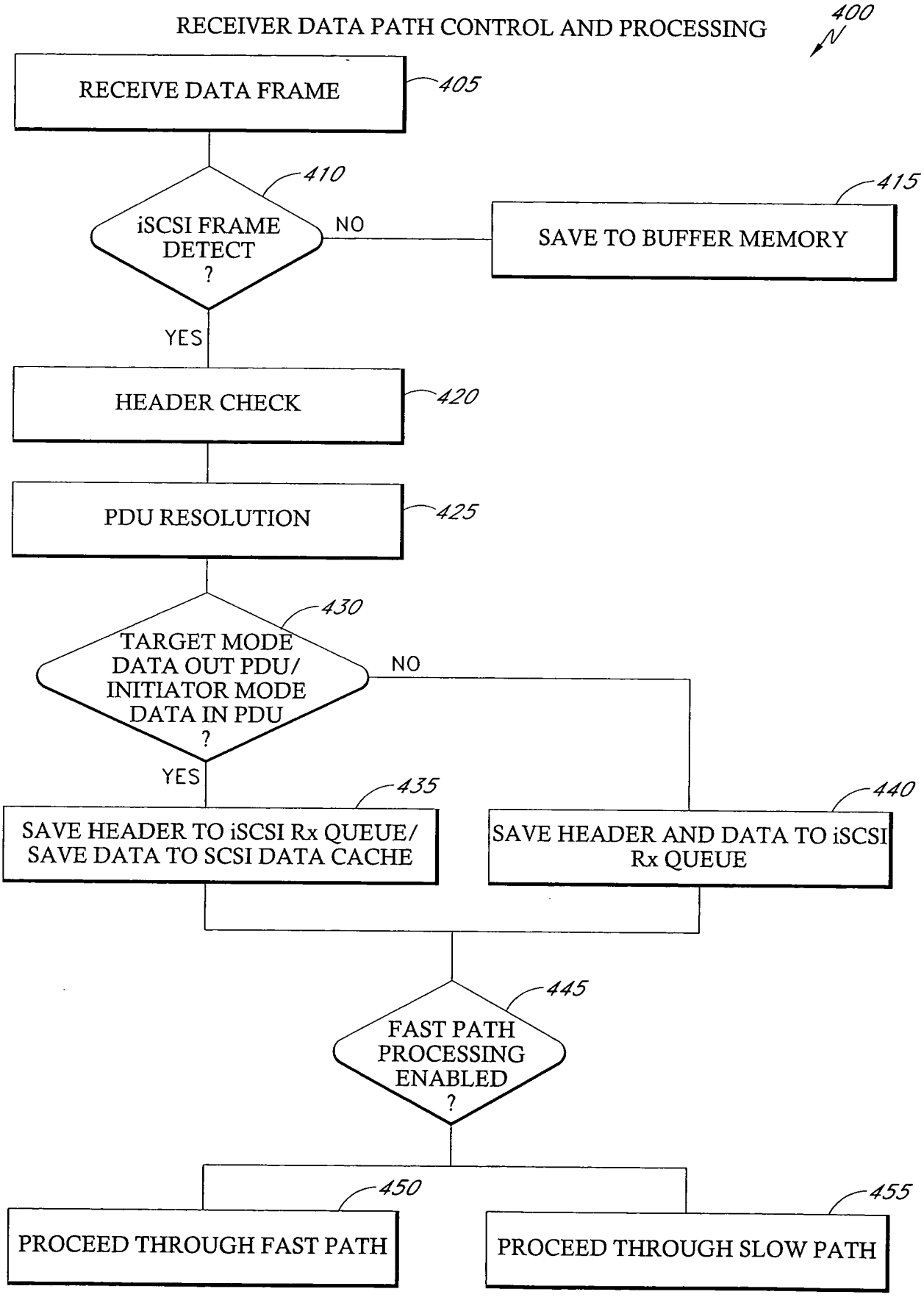

385
Non-Real Time Path
370
Real-Time Path
375
PIEMan Path
380
Transmit Processing
Hardware FSM
389
398
Application Processing
Software on CPU Complex
iSCSI Header Processing
Hardware FSM
392
iSCSI Protocol Processing
Software on CPU Complex
394
TCP Resequencing And Exception Processing
Software on NRT Processor
390
TCP Protocol Processing
Firmware on VLIW and Hardware Accelerators
387
iSCSI Header Processing
Hardware FSM
389
IP Exception and Management Processing
Software on PIEMan CPU
394
387
TCP Header Processing
Hardware FSM
389
IP Header Processing
Hardware FSM
389

FIG. 6C

FIG. 7


RECEIVER DATA PATH CONTROL AND PROCESSING
400
RECEIVE DATA FRAME
405
iSCSI FRAME DETECT ?
410
NO
SAVE TO BUFFER MEMORY
415
YES
HEADER CHECK
420
PDU RESOLUTION
425
TARGET MODE DATA OUT PDU/ INITIATOR MODE DATA IN PDU ?
430
NO
YES
SAVE HEADER TO iSCSI Rx QUEUE/ SAVE DATA TO SCSI DATA CACHE
435
SAVE HEADER AND DATA TO iSCSI Rx QUEUE
440
FAST PATH PROCESSING ENABLED ?
445
PROCEED THROUGH FAST PATH
450
PROCEED THROUGH SLOW PATH
455

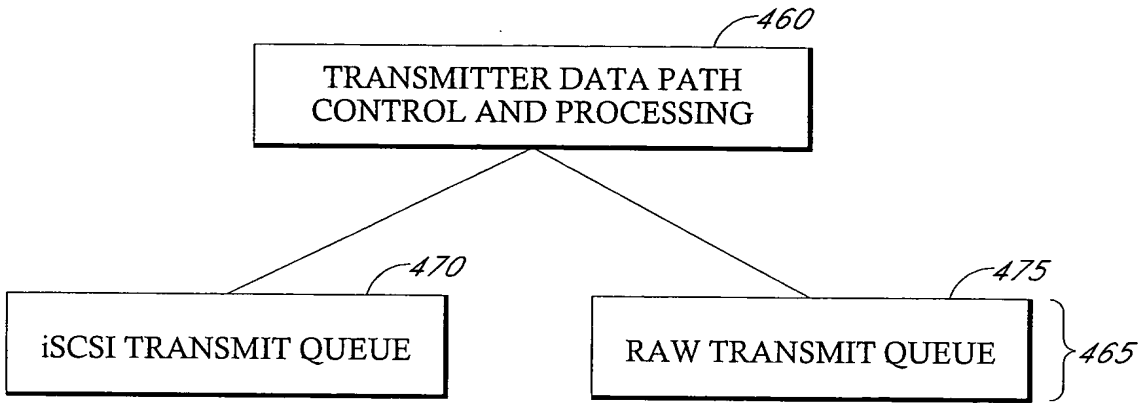

460
TRANSMITTER DATA PATH CONTROL AND PROCESSING
470
iSCSI TRANSMIT QUEUE
475
RAW TRANSMIT QUEUE
465

FIG. 8A

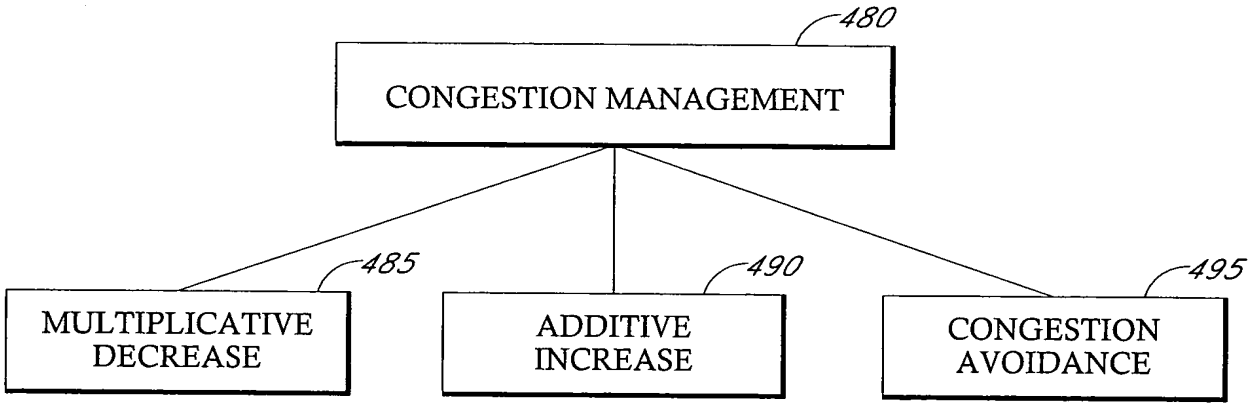

480
CONGESTION MANAGEMENT
485
MULTIPLICATIVE DECREASE
490
ADDITIVE INCREASE
495
CONGESTION AVOIDANCE

FIG. 8B

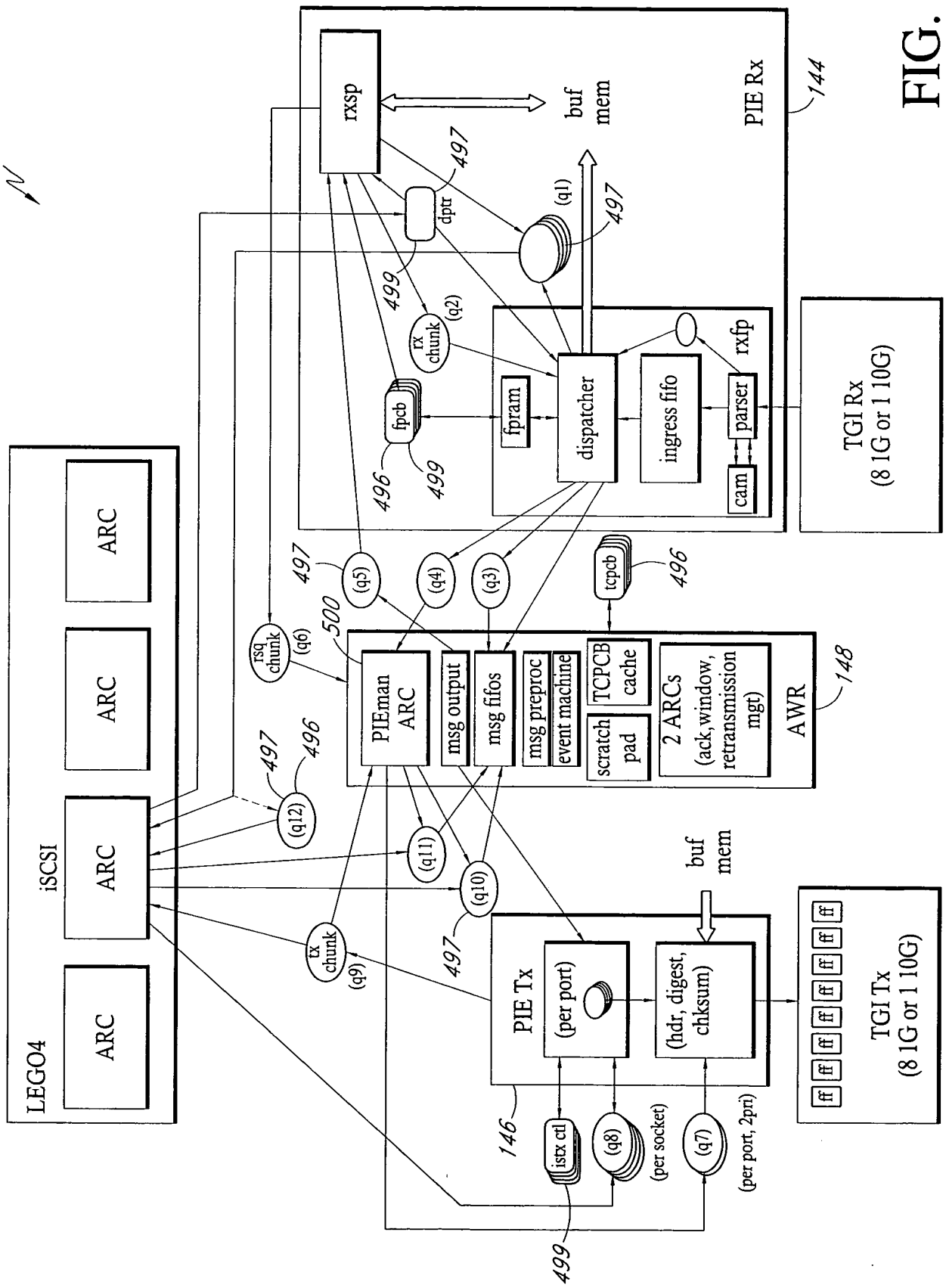

142
LEGO4
ARC
ARC
ARC
ARC
iSCSI
PIE Rx
144
rxsp
buf mem
dptr
(q1)
497
499
rx chunk
(q2)
fpcb
fpram
dispatcher
ingress fifo
rxfp
parser
cam
TGI Rx
(8 1G or 1 10G)
496
tcpcb
(q5)
(q4)
(q3)
rsq chunk
(q6)
500
PIEman ARC
msg output
msg fifos
msg preproc
event machine
TCPCB cache
scratch pad
2 ARCs (ack, window, retransmission mgt)
AWR
148
(q12)
(q11)
(q10)
tx chunk
(q9)
PIE Tx
(per port)
(hdr, digest, chksum)
TGI Tx
(8 1G or 1 10G)
146
istx ctl
(q8)
(per socket)
(q7)
(per port, 2pri)

FIG. 9

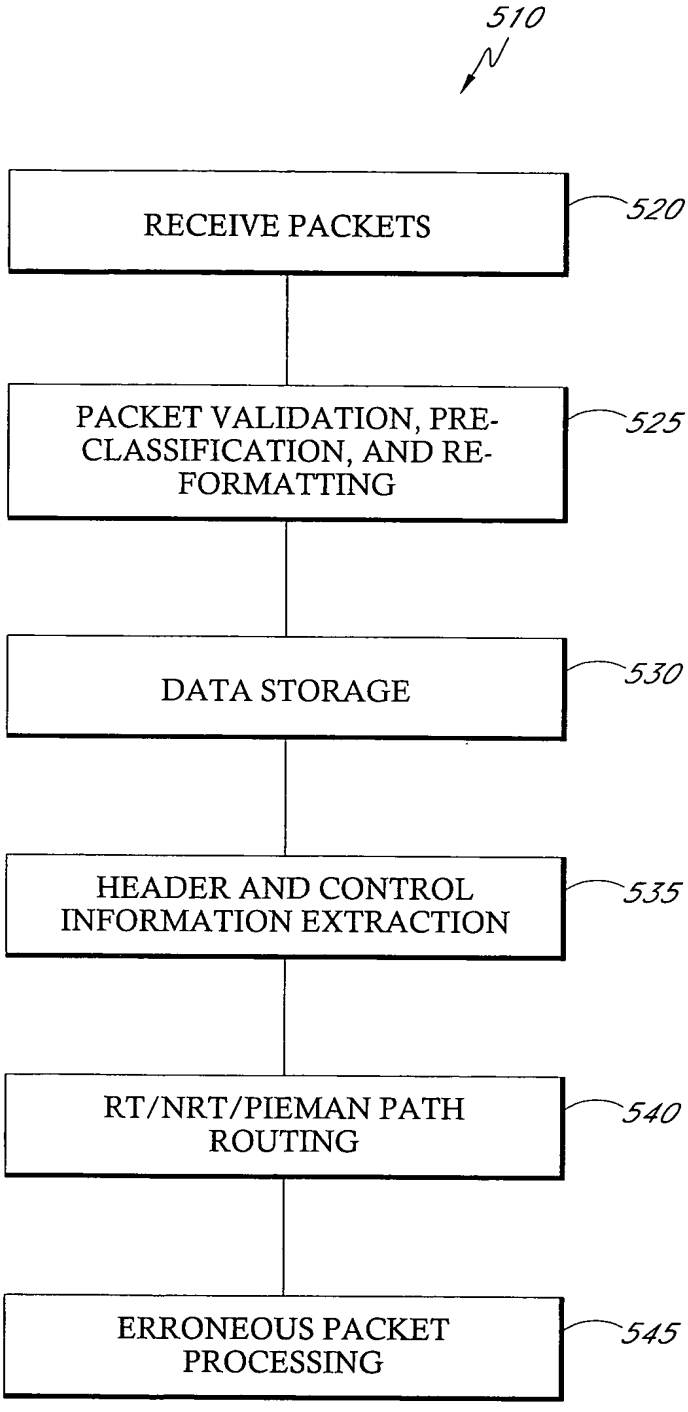

510
RECEIVE PACKETS
520
PACKET VALIDATION, PRE-CLASSIFICATION, AND RE-FORMATTING
525
DATA STORAGE
530
HEADER AND CONTROL INFORMATION EXTRACTION
535
RT/NRT/PIEMAN PATH ROUTING
540
ERRONEOUS PACKET PROCESSING
545

FIG. 10

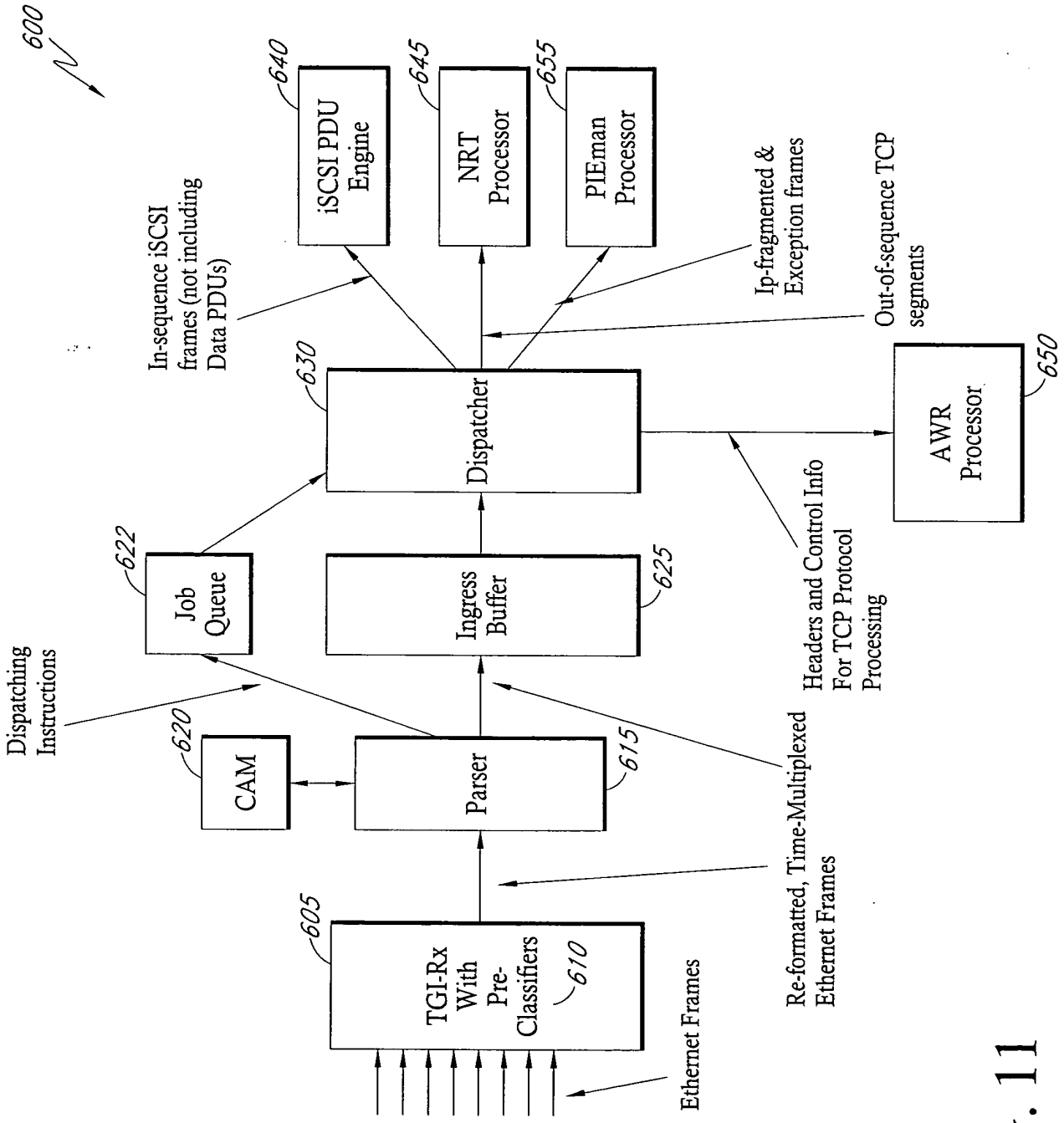

600
640
iSCSI PDU Engine
645
NRT Processor
655
PIEman Processor
In-sequence iSCSI frames (not including Data PDUs)
Ip-fragmented & Exception frames
Out-of-sequence TCP segments
630
Dispatcher
650
AWR Processor
622
Job Queue
Ingress Buffer
625
Dispatching Instructions
Headers and Control Info For TCP Protocol Processing
620
CAM
Parser
615
605
TGI-Rx With Pre-Classifiers
610
Re-formatted, Time-Multiplexed Ethernet Frames
Ethernet Frames

FIG. 11

TGI field	Description
tag[3:0]	Indicates the type of information in the dword, as preclassified by the TGI 0: invalid (interframe) 1: E (ethernet header) 2: EV (ethernet header, VLAN present) 3: E8 (ethernet header, 802.3 rfc1042 format) 4: EV8 (ethernet header, 802.3 rfc1042 format, VLAN present) 5: I (IP) 6: IO (IP options) 7: IF (IP fragmented) 8: T (TCP) 9: TO (TCP option) 10: U (UDP) 11: spare 12: S (iSCSI, i.e. TCP and DPORT or SPORT matches iSCSI 13: o (other, e.g. not IP, not TCP or UDP) 14: G (good EOF, dword holds checksum, frame length) 15: B (bad EOF, assert early if checksum or other error detected)
off[2:0]	Indicates the byte offset to tag boundary within dword, 0=left side

FIG. 12A

B0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

raw non-vlan and vlan frames

da						sa						type		ihl+	
tlen		i id		frag		i prot		cksm		sip				dip u	
dip l		sport		dport		seq num				ack num				hl/f	
win		cksm		urg											
da						sa						vlan		vtag	
type		ihl+		tlen		i id		frag		i prot		cksm		sip u	
sip l		dip				sport		dport		seq num				ack u	
ack l		hl/f		win		cksm		urg							
da						sa						vlan		vtag	
802.3 len, pat=ip								type		ihl+		tlen		i id	
frag		i prot		cksm		sip				dip				sport	
dport		seq num				ack u		ack l		hl/f		win		cksm	
urg															
da						sa						802.3 u			
802.3 l				type		ihl+		tlen		i id		frag		i prot	
cksm		sip				dip				sport		dport		seq u	
seq l		ack u		ack l		hl/f		win		cksm		urg			

B0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

raw non-vlan and vlan frames

tag,off															
e	0	da						sa					type	0	
i	0	ihl+		tlen		i id		frag		i prot		cksm	sip		
t	4	dip				sport		dport		seq num			ack num		
t	8	hl/f		win		cksm		urg							
ev	0	da						sa					type	vtag	
i	0	ihl+		tlen		i id		frag		i prot		cksm	sip		
t	4	dip				sport		dport		seq num			ack num		
t	8	hl/f		win		cksm		urg							
ev8	0	da						sa					type	vtag	
i	0	ihl+		tlen		i id		frag		i prot		cksm	sip		
t	4	dip				sport		dport		seq num			ack num		
t	8	hl/f		win		cksm		urg							
e8	0	da						sa					type	vtag	
i	0	ihl+		tlen		i id		frag		i prot		cksm	sip		
t	4	dip				sport		dport		seq num			ack num		
t	8	hl/f		win		cksm		urg							

FIG. 12B

#bits	Field	Description
4	State	State machine state
2	ts_offst	Timestamp offset (0=none, 1=22B from start of TCP header, 2=23B, 3=24B
1	t_left	TCP header starts in left half of dword
4	Reason	If nonzero, the reason from slow-path processing
1	Msw_par	Job FIFO MS word parity
1	d_left	1st dword stored (at next dword write qword to ingress fifo)
64	d_data	Stored dword
2	d_par	Stored dword parity
80		Total bits

FIG. 13A

Slow-Path Reason codes (*=set by dispatcher):

0 : nop -- fastpath iSCSI
1: ARP frame
2: other non-IP (not ARP) frame
3: IP fragment (if not, fragment zero could be iSCSI)
4: TCP but not iSCSI or runt iSCSI (flen<0x38)
5: UDP frame
6: ICMP frame
7: other IP frame (not IP fragment, TCP, UDP, or ICMP)
8: iSCSI, IP fragment zero
9: iSCSI, no socket ID found in CAM
a: iSCSI, unsupported option
*b: iSCSI, fastpath disabled
*c: iSCSI, out of sequence
*d: iSCSI, bad data boundary

FIG. 13B

CAM_LOAD I, data	=	Write CAM entry I with specified; Set Valid bit;
CAM_READ I	=	Read data contained in CAM entry I;
CAM_INV I	=	Clear the valid bit for CAM entry I;
CAM_REQ P	=	Initiate CAM search with Key elements written in the CAM-Key register for network port P;
CAM_RESULT P	=	Fetch result from CAM search for network port P;

FIG. 14

#bits	Field	Description
2	TS offset	If nonzero, byte offset minus 1 from start to TCP options to timestamp field
4	TCP option length	Size in wowrds of TCP options
4	slow-path Reason Code	If nonzero, packet takes slow path. Reason codes (*=set by dispatcher): 0 : nop -- fastpath iSCSI 1: ARP frame 2: other non-IP (not ARP) frame 3: IP fragment (if not, fragment zero could be iSCSI) 4: TCP but not iSCSI or runt iSCSI (flen<0x38) 5: UDP frame 6: ICMP frame 7: other IP frame (not IP fragment, TCP, UDP, or ICMP) 8: iSCSI, IP fragment zero 9: iSCSI, no socket ID found in CAM a: iSCSI, unsupported option *b: iSCSI, fastpath disabled *c: iSCSI, out of sequence *d: iSCSI, bad data boundary
1	ID valid	iSCSI socket ID valid
1	Init	Initiator mode
4	IP option length	Size in words of IP options
10	Socket ID	iSCSI socket ID
1	VLAN	16th byte contains Vlan tag (IP frame)
1	802.3	802.3 rfc 1042 coding was removed from ethernet header (IP frame)
14	frame length	Length of formatted frame in bytes
16	partial checksum	Checksum for UDP or partial TCP segment (info PIEman)

FIG. 15

#bits	field	description	notes	
58	job	jff output		
13	roffst	iff read offset	for random access of iff	
13	rckpt	iff read checkpoint (start of frame)	for calculating iff discard point (add flength)	
10	fctr	frame qword counter		
32	seq	TCP sequence number	if iSCSI (except flags)	for msgRxNotify and msgRxFrame, also need some job fields
32	ack	TCP acknowledgement number		
32	flgs	flags, TCP flags, TCP window size		
32	ts	TCP timestamp		
32	ets	TCP echo timestamp		
288	rcp0-rcp8	Rx chunk pointer 0 up to 8	depending on flength	for msgRxFrame
546		total (approx 69 bytes per port, total 552B)		

FIG. 16

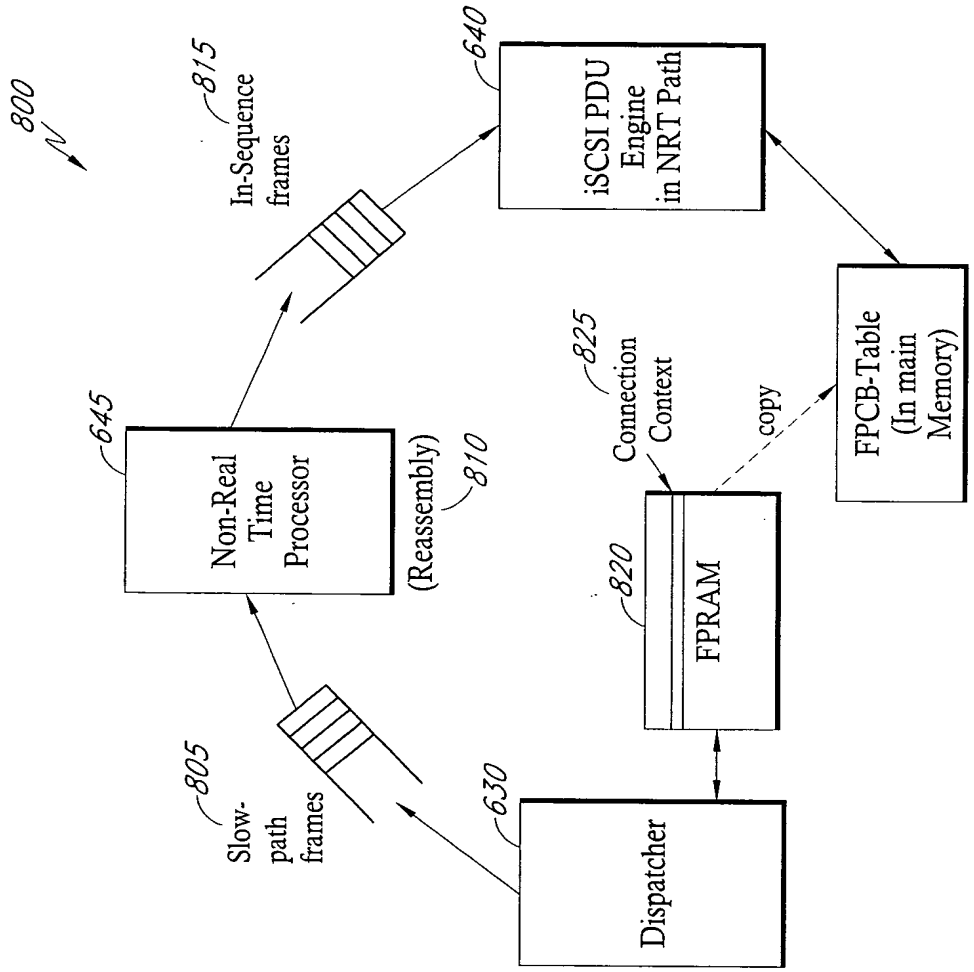

800
815
In-Sequence frames
640
iSCSI PDU Engine in NRT Path
825
Connection Context
copy
FPCB-Table (In main Memory)
645
Non-Real Time Processor
(Reassembly)
810
820
FPRAM
805
Slow-path frames
630
Dispatcher

FIG. 17

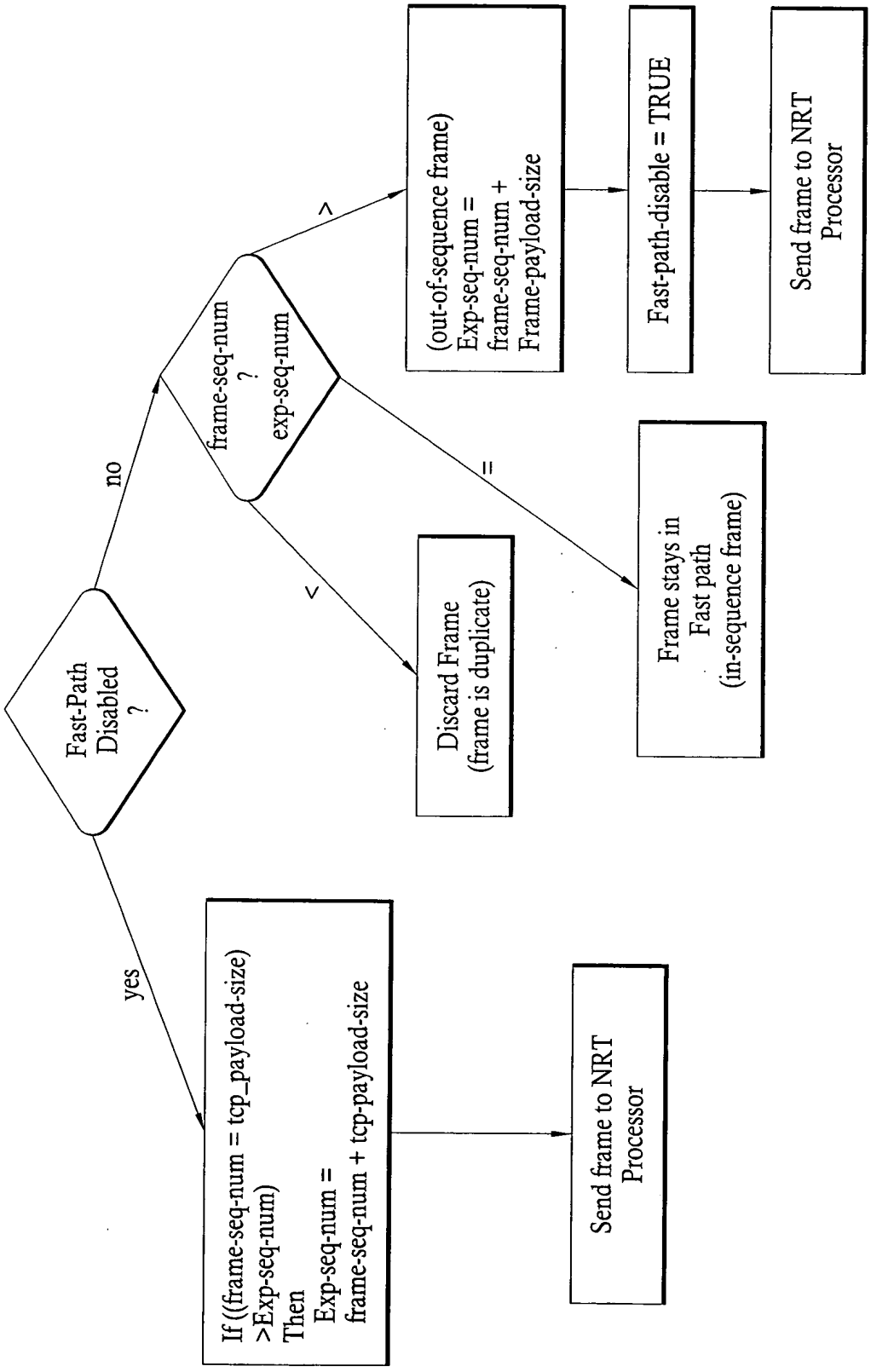

Fast-Path Disabled ?
yes
no
If ((frame-seq-num = tcp_payload-size) >Exp-seq-num) Then Exp-seq-num = frame-seq-num + tcp-payload-size
Send frame to NRT Processor
frame-seq-num ? exp-seq-num
>
(out-of-sequence frame) Exp-seq-num = frame-seq-num + Frame-payload-size
Fast-path-disable = TRUE
Send frame to NRT Processor
=
Frame stays in Fast path (in-sequence frame)
<
Discard Frame (frame is duplicate)

FIG. 18A

byte#	#bits	ddig_enb	field
0	8		Control bits:
		sp	7: slow path forever (disallows automatic return to fast path)
		lo_pri	6: low priority (0=high pri). If set and awr_prx_rdy_lo false, discard frame
		ts_val	5: TCP timestamp valid. If invalid, timestamp check automatically passes
		rx_dis	4: PDU Rx disable. Do not store subsequent data or header segments
		flush	3: PDU flush. Do not store data segment, auto-reset flush at PDU end
		ddig_enb	2: data digest enable. Enables check of iSCSI data segment CRC
		hdig_enb	1: header digest enable. Enables check of iSCSI header segment CRC
		fp_enb	0: fastpath enable. If disabled, entire frame stored to Rx chunk.
1-4	32	nxt_seq	Next TCP sequence number expected
5	8	ts	TCP timestamp [17:10]
6-9	32	pcrc	iSCSI partial digest (checked so far)
10-12	24	dsctl	iSCSI ata segment control:
	(1)	wenb	23: write SCSI data to buffer memory, vs header/non-SCSI data to Rx queue (flags how to interpret bytes 17-31)
	(23)	dsctr	22-0: data segment down counter (# words remaining in data seg, including data digest if present)
13-16	32	wptr	SCSI data write pointer ([3:0] indicates #qword residual bytes)
17-31	120		Within iSCSI header segments:
(17)	(8)	hctl	[7-4]=spare, [3]=final PDU, [2]=scsi data, [1:0]=#residual bytes
(18)	(8)	state	[7-5]=spare, [4-0]= state
(19-21)	(24)	wres	Word residual (up to 3 bytes)
(22)	(8)	ahctr	Additional header segment down counter (#words in AHSs)
(23-26)	(32)	hoffst	Data offset from data out or data in header
(27-28)	(16)	dplen	Data length from DPT
(29-31)	(24)	dpoffst	Data offset from DPT, bits 31:24 (sb 512B boundary)
			Within SCSi data segments:
17-31	120	qres	Qword residual (up to 15 bytes)

FIG. 18B

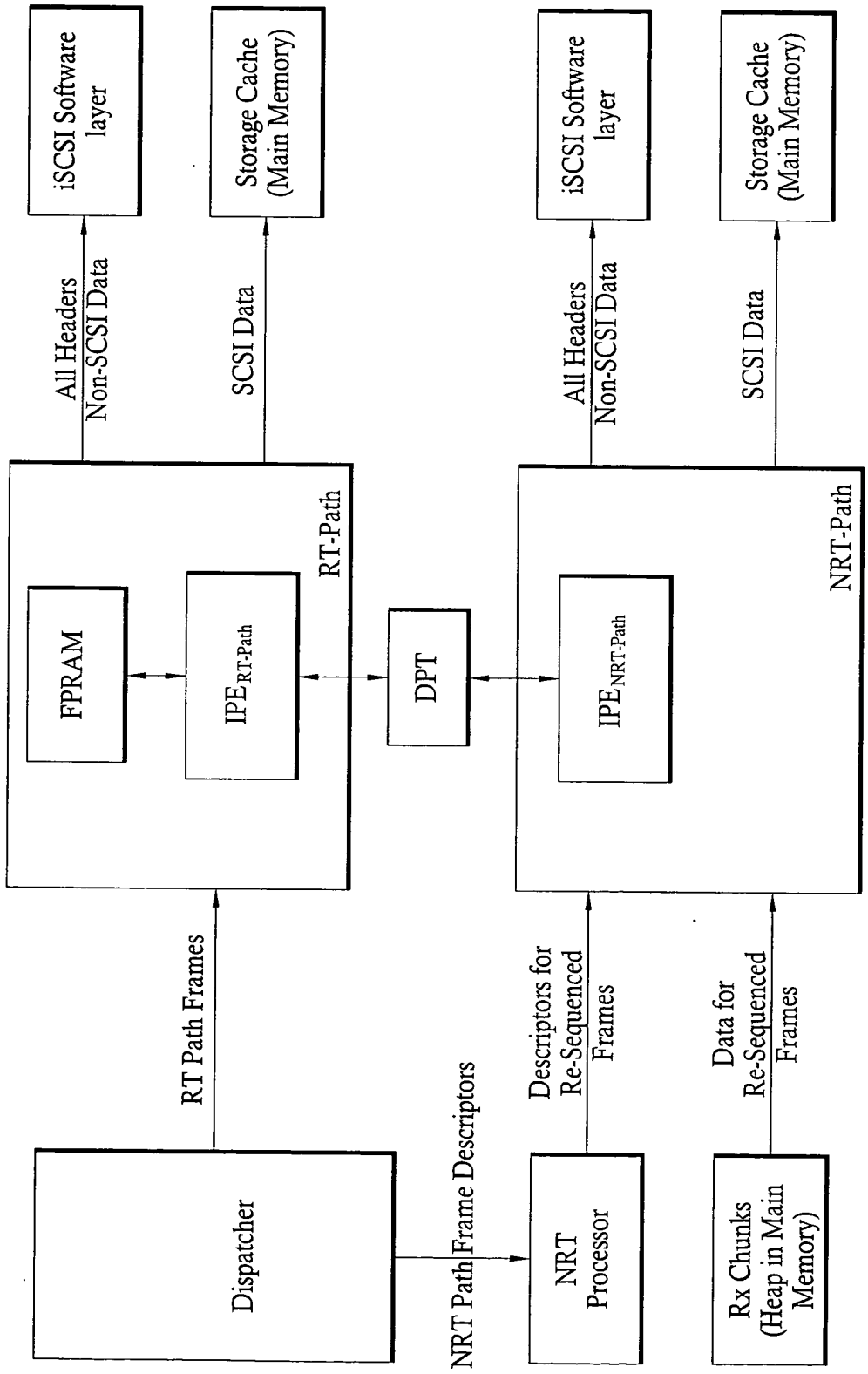

iSCSI Software layer
Storage Cache (Main Memory)
iSCSI Software layer
Storage Cache (Main Memory)
All Headers Non-SCSI Data
SCSI Data
All Headers Non-SCSI Data
SCSI Data
RT-Path
NRT-Path
FPRAM
IPE RT-Path
DPT
IPE NRT-Path
RT Path Frames
NRT Path Frame Descriptors
Descriptors for Re-Sequenced Frames
Data for Re-Sequenced Frames
Dispatcher
NRT Processor
Rx Chunks (Heap in Main Memory)

FIG. 18C

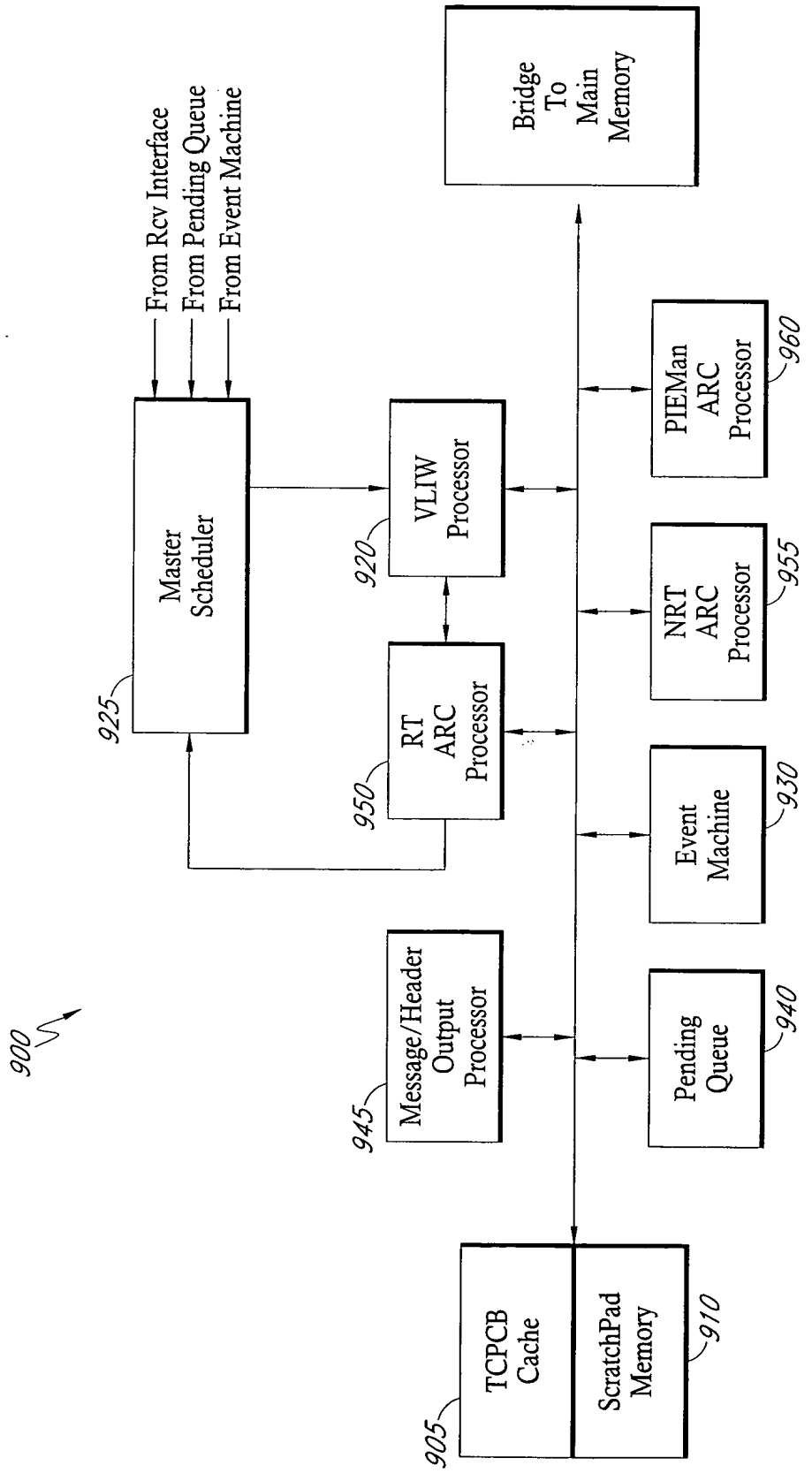

900
From Rcv Interface
From Pending Queue
From Event Machine
Bridge To Main Memory
Master Scheduler
925
VLIW Processor
920
RT ARC Processor
950
PIEMan ARC Processor
960
NRT ARC Processor
955
Event Machine
930
Pending Queue
940
Message/Header Output Processor
945
TCPCB Cache
ScratchPad Memory
910
905

FIG. 19

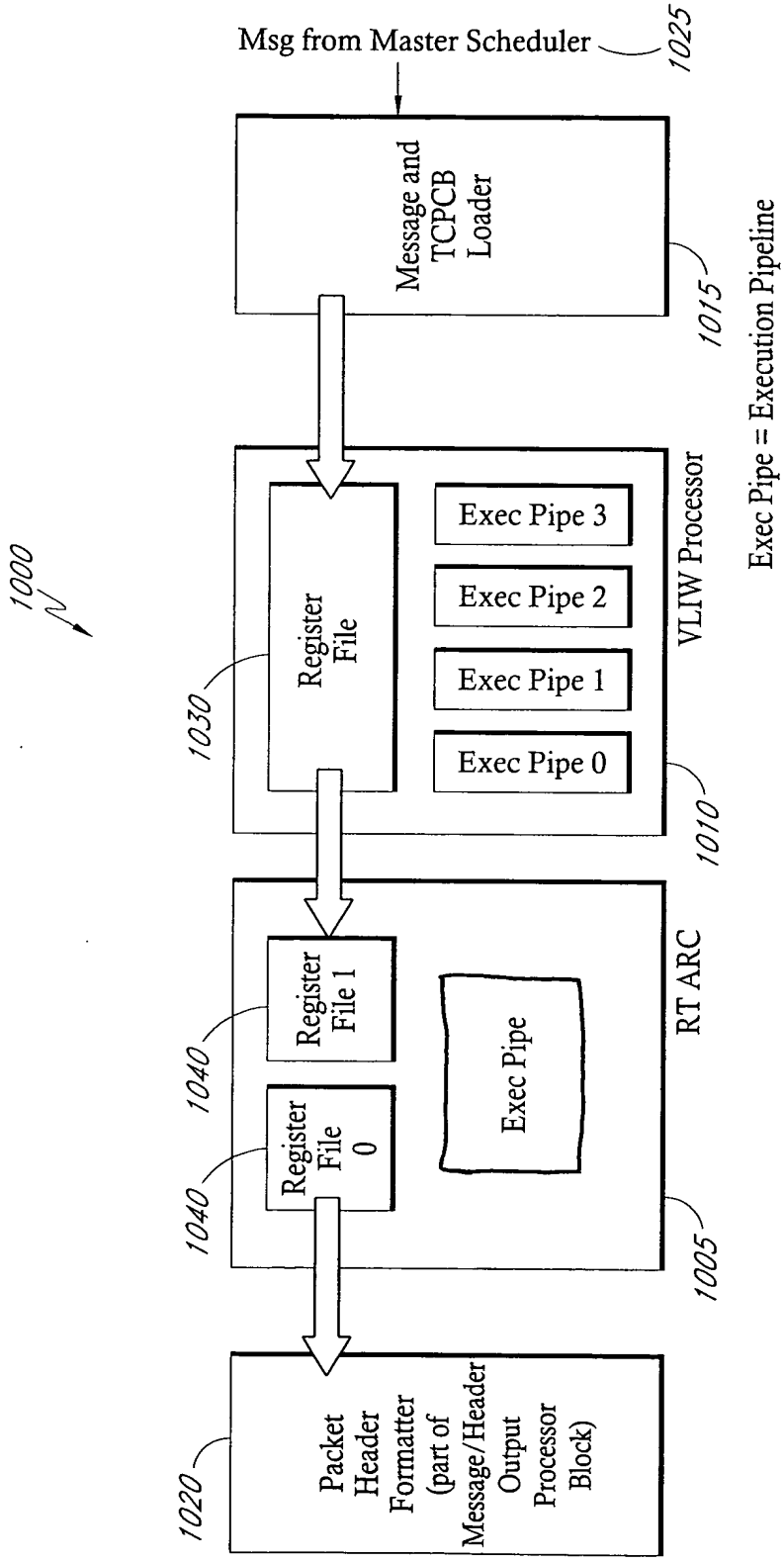

1000
Msg from Master Scheduler
1025
Message and TCPCB Loader
1015
Exec Pipe = Execution Pipeline
Register File
1030
Exec Pipe 3
Exec Pipe 2
Exec Pipe 1
Exec Pipe 0
VLIW Processor
1010
Register File 1
1040
Register File 0
1040
Exec Pipe
RT ARC
1005
Packet Header Formatter (part of Message/Header Output Processor Block)
1020

FIG. 20

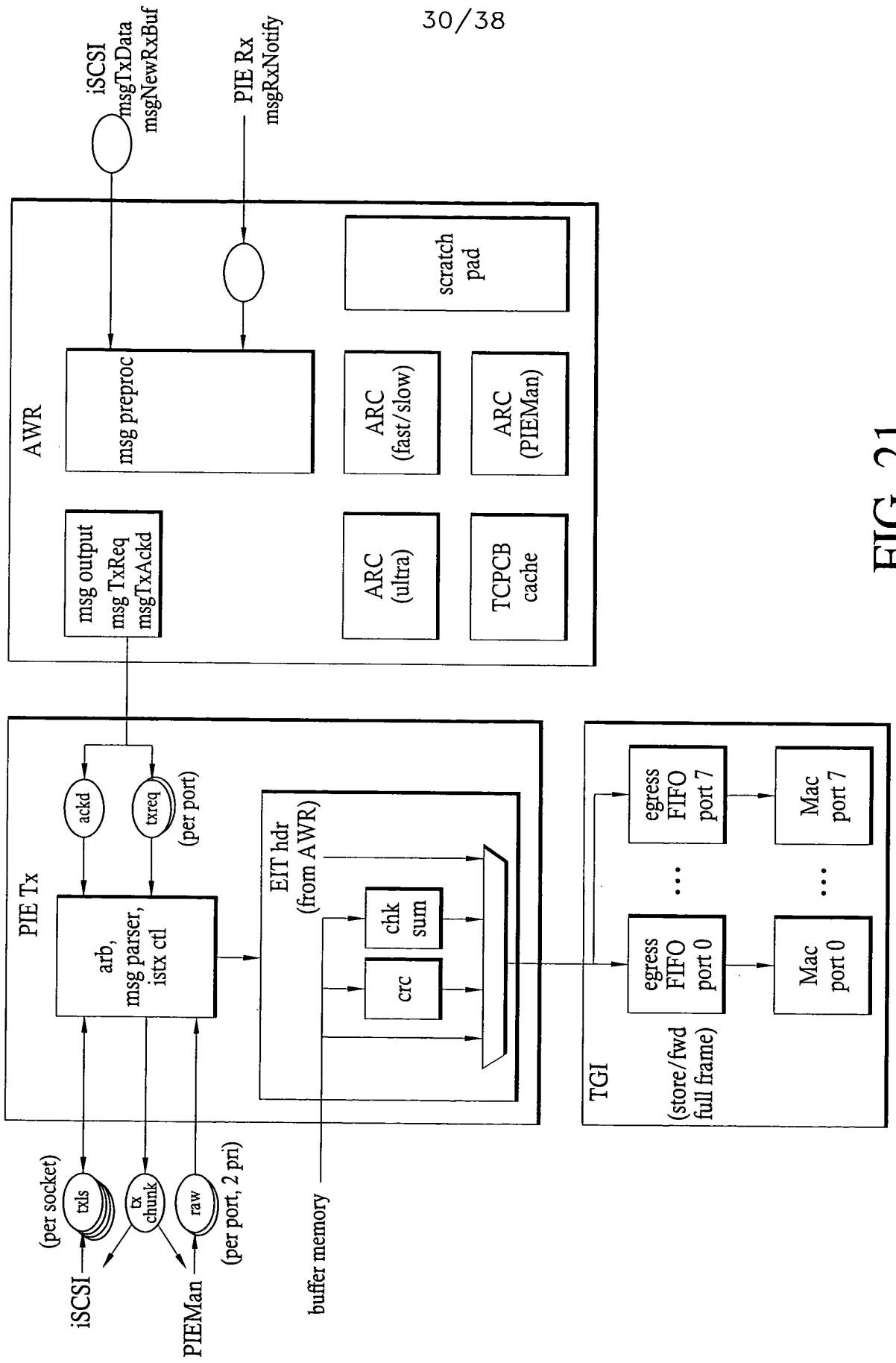

iSCSI
msgTxData
msgNewRxBuf
PIE Rx
msgRxNotify
AWR
msg preproc
scratch pad
ARC (fast/slow)
ARC (PIEMan)
msg output
msg TxReq
msgTxAckd
ARC (ultra)
TCPCB cache
PIE Tx
ackd
txreq
(per port)
arb, msg parser, istx ctl
EIT hdr (from AWR)
chk sum
crc
TGI
egress FIFO port 7
egress FIFO port 0
Mac port 7
Mac port 0
(store/fwd full frame)
(per socket)
txls
tx chunk
raw
(per port, 2 pri)
iSCSI
PIEMan
buffer memory

FIG. 21

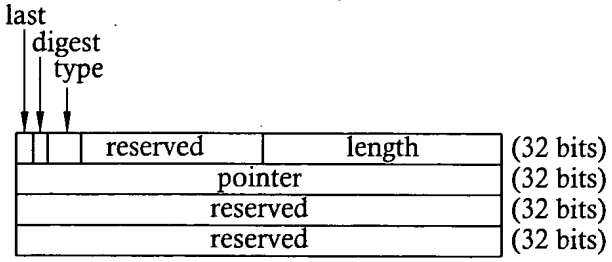


FIG. 22

data in PDU

0	1	h	rsv	48	ptr to hdr in Tx ch	reserved	reserved
0	1	d	rsv	#bytes 1	ptr to data1	reserved	reserved
0	1	d	rsv	#bytes 2	ptr to data2	reserved	reserved
1	1	d	rsv	#bytes 3	ptr to data3	reserved	reserved

login response PDU

0	1	h	rsv	48	ptr to hdr in Tx ch	reserved	reserved
0	1	t	rsv	#bytes	ptr to text	reserved	reserved

FIG. 23

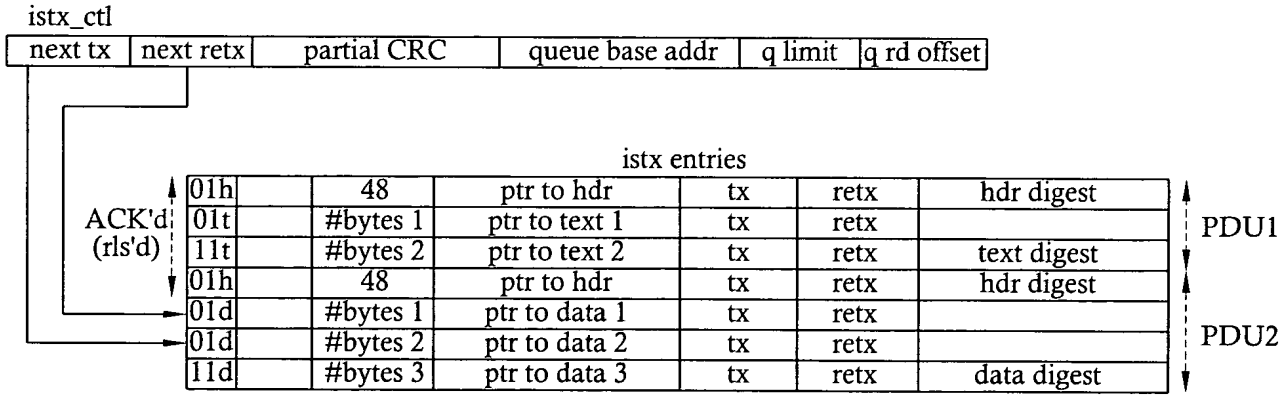


FIG. 24

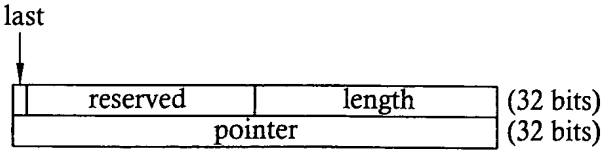


FIG. 25

Command	Description
Push	Writes data beginning at write pointer and saves new write pointer in descriptor.
Push/Inc	Writes data beginning at write pointer, increments counter field by one, and saves new write pointer and counter in descriptor
Inc	Increments counter field by a specified amount and saves new counter in descriptor
Inc Bytes	Increments write pointer field by a special amount and saves new write pointer in descriptor
Push/Chkpt	Writes data beginning at write pointer and saves new write pointer in descriptor and in descriptor extension as write checkpoint.
Push/Inc/Chkpt	Writes data beginning at write pointer, increments counter field by one, saves new write pointer and counter in descriptor, and saves new write pointer in descriptor extension as write checkpoint.
Inc/Chkpt	Increments counter field by a specified amount, saves new counter in descriptor, and copies current write pointer to write checkpoint
Rewind	Copies write checkpoint to write pointer and saves result in descriptor
Peek	Reads data beginning at read pointer (for queues) or write pointer (for stacks) but does not save new pointer in descriptor
Pop	Reads data beginning at read pointer (for queues) or write pointer (for stacks) but saves new pointer in descriptor
Pop/Dec	Reads data beginning at read pointer (for queues) or write pointer (for stacks) decrements counter field by one, and saves new pointer in descriptor
Dec	Decrements counter field by a specified amount and saves new counter in descriptor
Dec Bytes	Decrements read pointer (for queues) or write pointer (for stacks) by a specified amount and saves new pointer in descriptor

FIG. 26

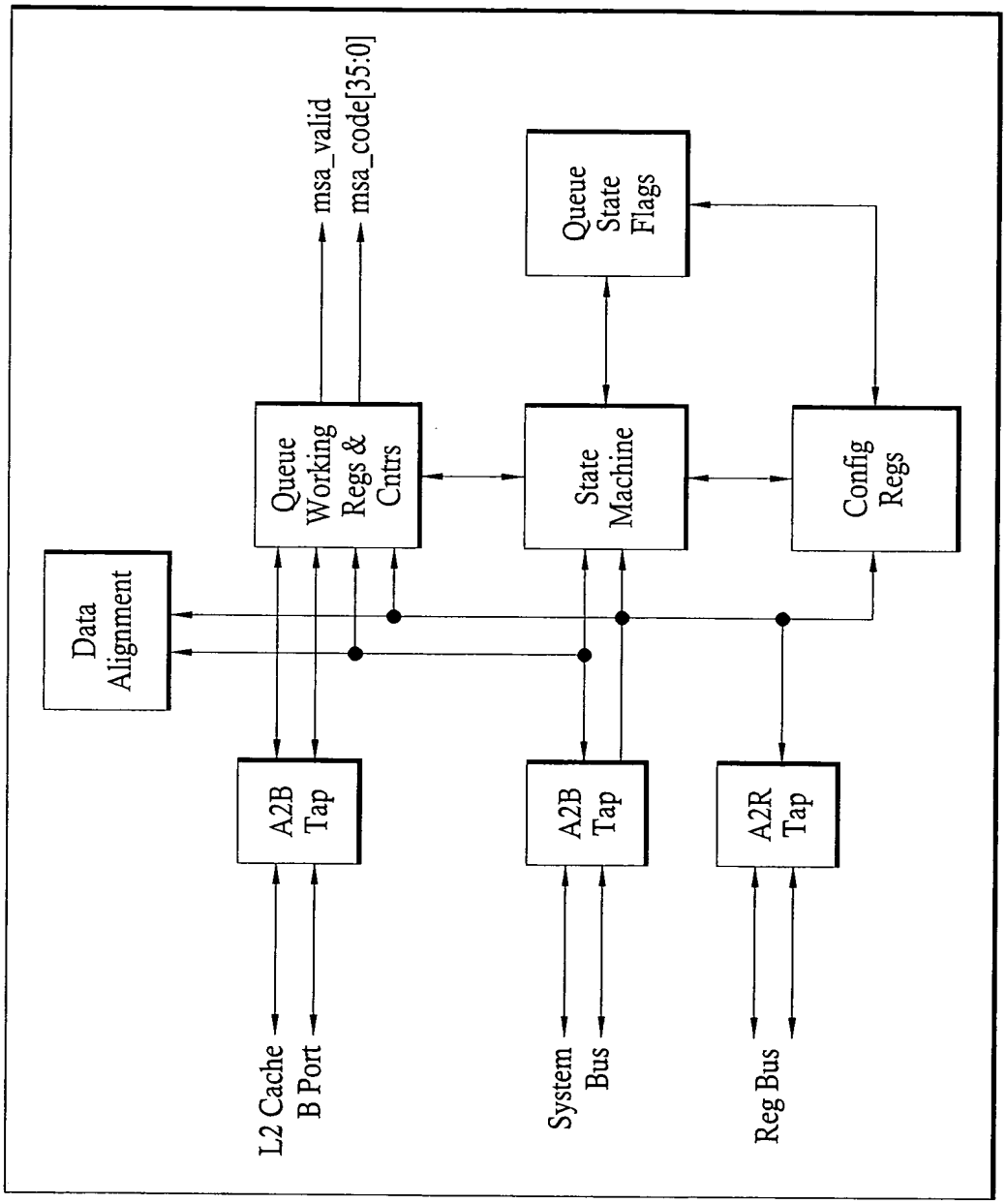

msa_valid
msa_code[35:0]
Queue State Flags
Queue Working Regs & Cntrs
State Machine
Config Regs
Data Alignment
A2B Tap
A2B Tap
A2R Tap
L2 Cache B Port
System Bus
Reg Bus

FIG. 27

00000	---	01000	---	10000	---	11000	Rewind
00001	Pop/Dec/ (Read)	01001	Pop	10001		11001	Peek
00010	Push/Inc (Write)	01010	Push	10010	Push/Inc/Chkpt	11010	Push/Chkpt
00011	---	01011	Dec	10011	---	11011	---
00100	---	01100	---	10100	---	11100	---
00101	---	01101	---	10101	---	11101	---
00110	---	01110	Inc	10110	Inc/Chkpt	11110	---
00111	---	01111	---	10111	---	11111	---

FIG. 28

00	Queue Not Empty	10	Queue Underflow
01	Queue Empty	11	Not used

FIG. 29

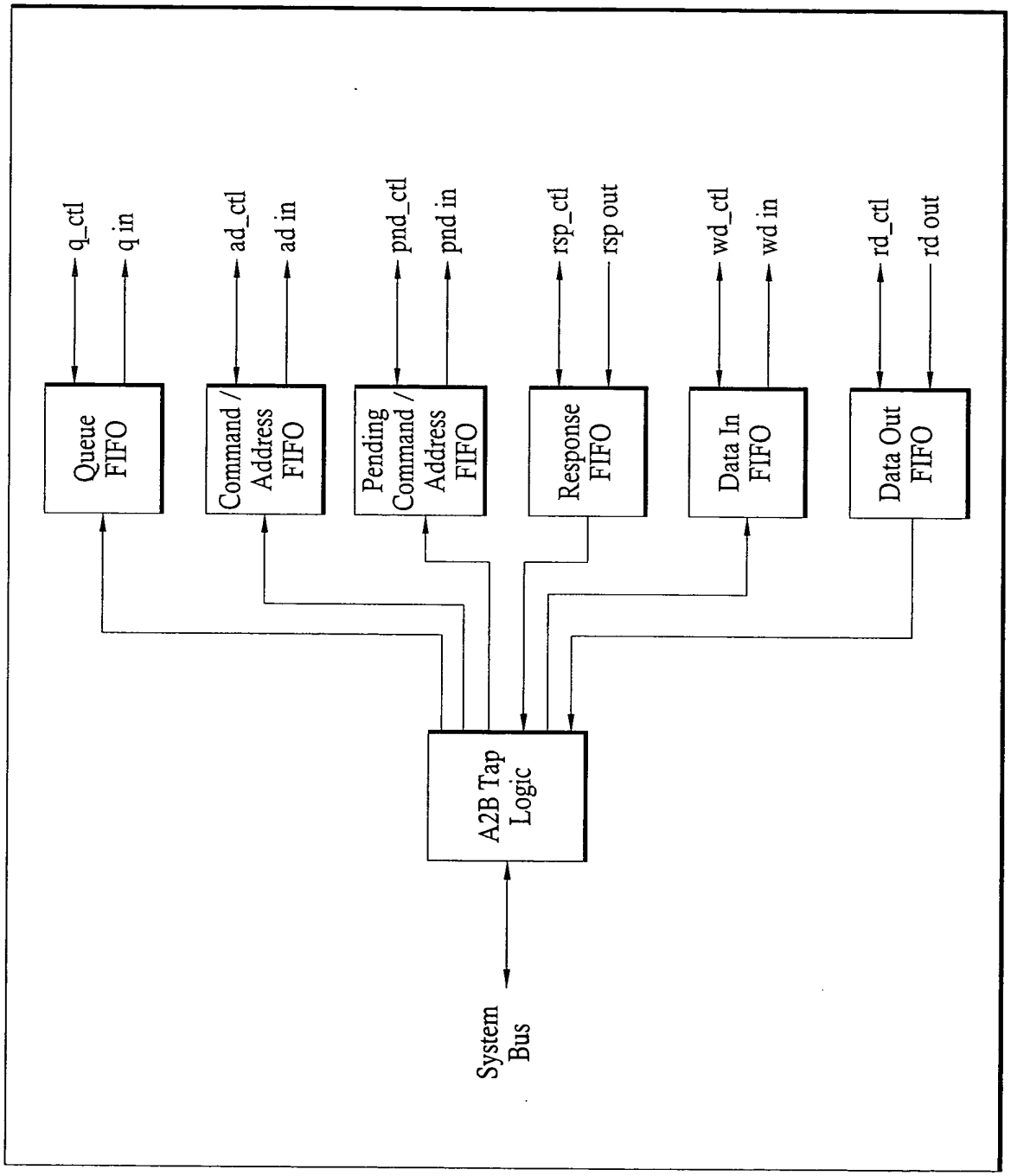

q_ctl
q in
ad_ctl
ad in
pnd_ctl
pnd in
rsp_ctl
rsp out
wd_ctl
wd in
rd_ctl
rd out
Queue FIFO
Command / Address FIFO
Pending Command / Address FIFO
Response FIFO
Data In FIFO
Data Out FIFO
A2B Tap Logic
System Bus

FIG. 30

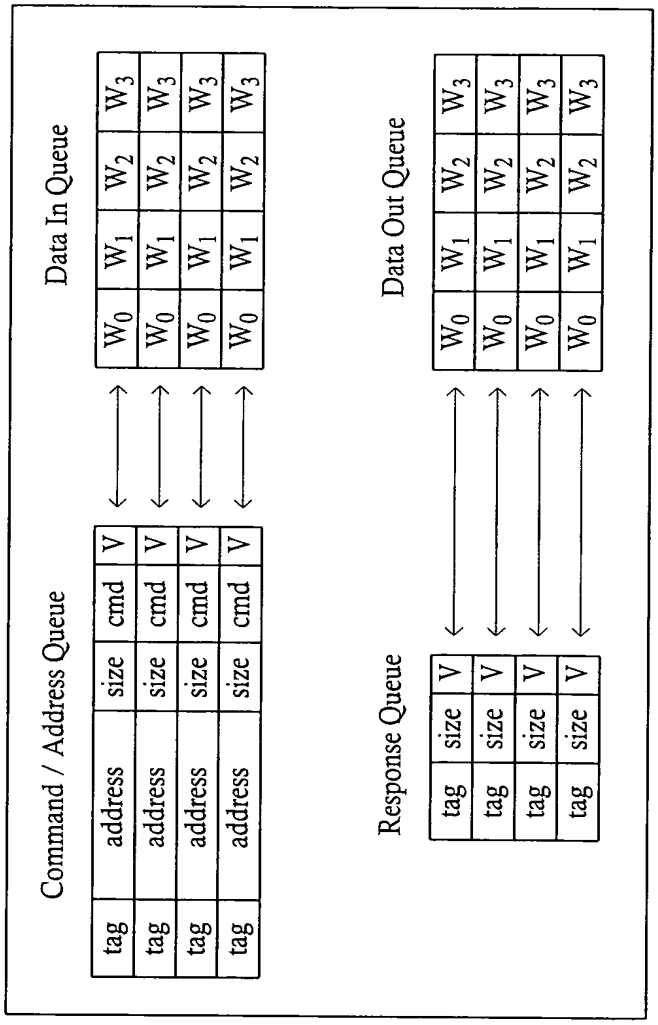
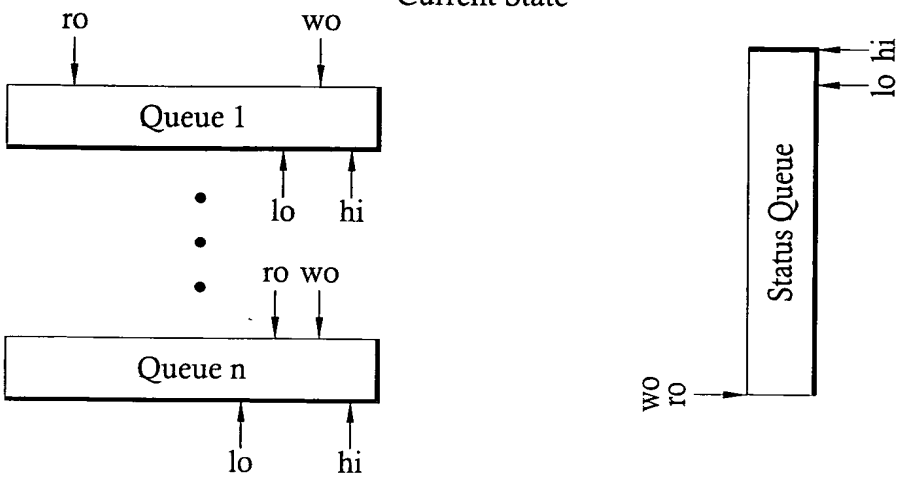
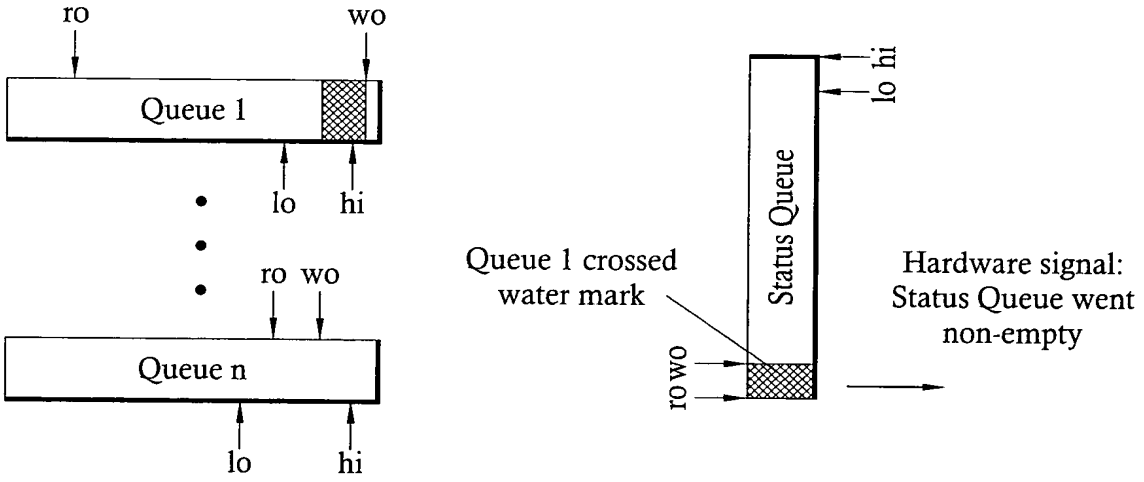

Command / Address Queue
tag
address
size
cmd
V
Data In Queue
W_0
W_1
W_2
W_3
Data Out Queue
Response Queue

FIG. 31

Current State



Push to Queue 1



Pop from Queue n

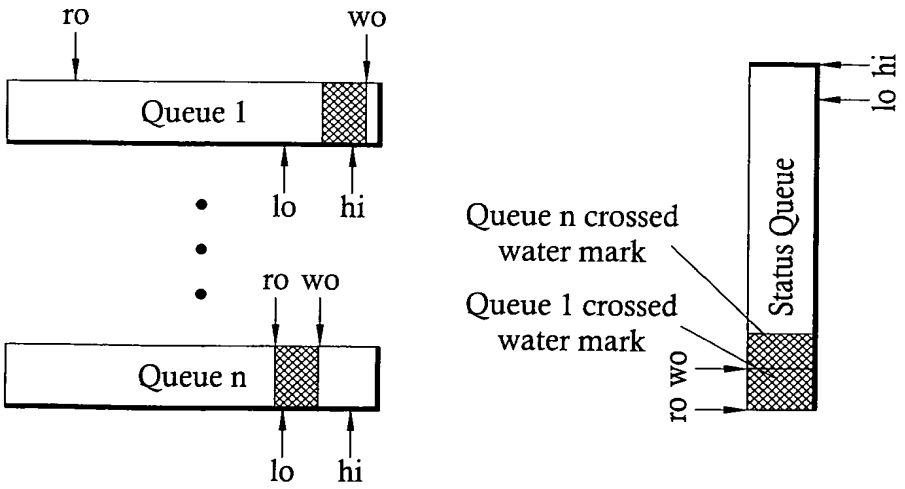


FIG. 32

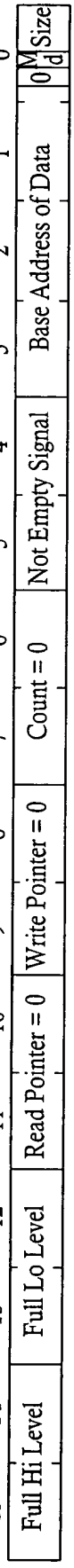

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
Full Hi Level
Full Lo Level
Read Pointer = 0
Write Pointer = 0
Count = 0
Not Empty Signal
Base Address of Data
0
Md
Size

FIG. 33

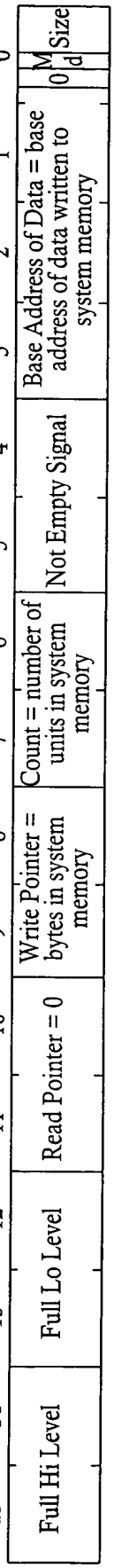

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
Full Hi Level
Full Lo Level
Read Pointer = 0
Write Pointer = bytes in system memory
Count = number of units in system memory
Not Empty Signal
Base Address of Data = base address of data written to system memory
0
Md
Size

FIG. 34